

46th International Symposium on Mathematical Foundations of Computer Science

MFCS 2021, August 23–27, 2021, Tallinn, Estonia

Edited by

Filippo Bonchi

Simon J. Puglisi



Editors

Filippo Bonchi 

University of Pisa, Italy
filippo.bonchi@unipi.it

Simon J. Puglisi 

University of Helsinki, Finland
simon.puglisi@helsinki.fi

ACM Classification 2012

Theory of computation

ISBN 978-3-95977-201-3

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-201-3>.

Publication date

August, 2021

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0): <https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.MFCS.2021.0

ISBN 978-3-95977-201-3

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

LIPICs – Leibniz International Proceedings in Informatics

LIPICs is a series of high-quality conference proceedings across all fields in informatics. LIPICs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Luca Aceto (*Chair*, Reykjavik University, IS and Gran Sasso Science Institute, IT)
- Christel Baier (TU Dresden, DE)
- Mikolaj Bojanczyk (University of Warsaw, PL)
- Roberto Di Cosmo (Inria and Université de Paris, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Daniel Král' (Masaryk University - Brno, CZ)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN)
- Anca Muscholl (University of Bordeaux, FR)
- Chih-Hao Luke Ong (University of Oxford, GB)
- Phillip Rogaway (University of California, Davis, US)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Raimund Seidel (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

■ Contents

Preface	
<i>Filippo Bonchi and Simon J. Puglisi</i>	0:xi
Conference Organization	
.....	0:xiii

Invited Talks

Non-Axiomatizability of the Equational Theories of Positive Relation Algebras	
<i>Amina Doumane</i>	1:1–1:1
A Deep Dive into the Weisfeiler-Leman Algorithm	
<i>Martin Grohe</i>	2:1–2:1
Holonomic Techniques, Periods, and Decision Problems	
<i>Joël Ouaknine</i>	3:1–3:1
On Dynamic Graphs	
<i>Eva Rotenberg</i>	4:1–4:1
Sublinear Algorithms for Edit Distance	
<i>Barna Saha</i>	5:1–5:1

Regular Papers

An Approximation Algorithm for the Matrix Tree Multiplication Problem	
<i>Mahmoud Abo-Khamis, Ryan Curtin, Sungjin Im, Benjamin Moseley, Hung Ngo, Kirk Pruhs, and Alireza Samadian</i>	6:1–6:14
Depth-First Search in Directed Planar Graphs, Revisited	
<i>Eric Allender, Archit Chauhan, and Samir Datta</i>	7:1–7:22
Order Reconfiguration Under Width Constraints	
<i>Emmanuel Arrighi, Henning Fernau, Mateus de Oliveira Oliveira, and Petra Wolf</i>	8:1–8:15
Universal Gauge-Invariant Cellular Automata	
<i>Pablo Arrighi, Marin Costes, and Nathanaël Eon</i>	9:1–9:14
Equivalence Testing of Weighted Automata over Partially Commutative Monoids	
<i>V. Arvind, Abhranil Chatterjee, Rajit Datta, and Partha Mukhopadhyay</i>	10:1–10:15
Finitely Tractable Promise Constraint Satisfaction Problems	
<i>Kristina Asimi and Libor Barto</i>	11:1–11:16
A Generic Strategy Improvement Method for Simple Stochastic Games	
<i>David Auger, Xavier Badin de Montjoye, and Yann Strozecki</i>	12:1–12:22
(Un)Decidability for History Preserving True Concurrent Logics	
<i>Paolo Baldan, Alberto Carraro, and Tommaso Padoan</i>	13:1–13:16

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Parameterized Complexity of Feature Selection for Categorical Data Clustering <i>Sayan Bandyopadhyay, Fedor V. Fomin, Petr A. Golovach, and Kirill Simonov</i> ..	14:1–14:14
Decision Questions for Probabilistic Automata on Small Alphabets <i>Paul C. Bell and Pavel Semukhin</i>	15:1–15:17
Ideal Membership Problem for Boolean Minority and Dual Discriminator <i>Arpitha P. Bharathi and Monaldo Mastrolilli</i>	16:1–16:20
Graph Traversals as Universal Constructions <i>Siddharth Bhaskar and Robin Kaarsgaard</i>	17:1–17:20
Space-Efficient Fault-Tolerant Diameter Oracles <i>Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck</i>	18:1–18:16
ω -Forest Algebras and Temporal Logics <i>Achim Blumensath and Jakub Lédl</i>	19:1–19:21
Constructing Deterministic ω -Automata from Examples by an Extension of the RPNI Algorithm <i>León Bohn and Christof Löding</i>	20:1–20:18
Computational Complexity of Covering Multigraphs with Semi-Edges: Small Cases <i>Jan Bok, Jiří Fiala, Petr Hliněný, Nikola Jedličková, and Jan Kratochvíl</i>	21:1–21:15
Coherent Control and Distinguishability of Quantum Channels via PBS-Diagrams <i>Cyril Branciard, Alexandre Clément, Mehdi Mhalla, and Simon Perdrix</i>	22:1–22:20
Reconfiguring Independent Sets on Interval Graphs <i>Marcin Briański, Stefan Felsner, Jędrzej Hodor, and Piotr Micek</i>	23:1–23:14
Finite Convergence of μ -Calculus Fixpoints on Genuinely Infinite Structures <i>Florian Bruse, Marco Sälzer, and Martin Lange</i>	24:1–24:19
Dots & Boxes Is PSPACE-Complete <i>Kevin Buchin, Mart Hagedoorn, Irina Kostitsyna, and Max van Mulken</i>	25:1–25:18
Uncertain Curve Simplification <i>Kevin Buchin, Maarten Löffler, Aleksandr Popov, and Marcel Roeloffzen</i>	26:1–26:22
Fractional Homomorphism, Weisfeiler-Leman Invariance, and the Sherali-Adams Hierarchy for the Constraint Satisfaction Problem <i>Silvia Butti and Víctor Dalmau</i>	27:1–27:19
A Decidable Equivalence for a Turing-Complete, Distributed Model of Computation <i>Arnaldo Cesco and Roberto Gorrieri</i>	28:1–28:18
Black-Box Hypotheses and Lower Bounds <i>Brynmor K. Chapman and R. Ryan Williams</i>	29:1–29:22
Geometry of Interaction for ZX-Diagrams <i>Kostia Chardonnet, Benoît Valiron, and Renaud Vilmart</i>	30:1–30:16
Diameter Versus Certificate Complexity of Boolean Functions <i>Siddhesh Chaubal and Anna Gál</i>	31:1–31:22

Budgeted Dominating Sets in Uncertain Graphs <i>Keerti Choudhary, Avi Cohen, N. S. Narayanaswamy, David Peleg, and R. Vijayaragunathan</i>	32:1–32:22
On the Complexity of the Escape Problem for Linear Dynamical Systems over Compact Semialgebraic Sets <i>Julian D’Costa, Engel Lefaucheur, Eike Neumann, Joël Ouaknine, and James Worrell</i>	33:1–33:21
The Pseudo-Skolem Problem is Decidable <i>Julian D’Costa, Toghrul Karimov, Rupak Majumdar, Joël Ouaknine, Mahmoud Salamati, Sadegh Soudjani, and James Worrell</i>	34:1–34:21
A Recursion-Theoretic Characterization of the Probabilistic Class PP <i>Ugo Dal Lago, Reinhard Kahle, and Isabel Oitavem</i>	35:1–35:12
Parallel Polynomial Permanent Mod Powers of 2 and Shortest Disjoint Cycles <i>Samir Datta and Kishlaya Jaiswal</i>	36:1–36:22
On the Relative Power of Linear Algebraic Approximations of Graph Isomorphism <i>Anuj Dawar and Danny Vagnozzi</i>	37:1–37:16
Maximum Cut on Interval Graphs of Interval Count Four Is NP-Complete <i>Celina M. H. de Figueiredo, Alexander A. de Melo, Fabiano S. Oliveira, and Ana Silva</i>	38:1–38:15
Fuzzy Simultaneous Congruences <i>Max A. Deppert, Klaus Jansen, and Kim-Manuel Klein</i>	39:1–39:16
Pebble Transducers with Unary Output <i>Gaëtan Douéneau-Tabot</i>	40:1–40:17
Graph Characterization of the Universal Theory of Relations <i>Amina Doumane</i>	41:1–41:15
Co-Degeneracy and Co-Treewidth: Using the Complement to Solve Dense Instances <i>Gabriel L. Duarte, Mateus de Oliveira Oliveira, and Uéverton S. Souza</i>	42:1–42:17
Isometric Embeddings in Trees and Their Use in Distance Problems <i>Guillaume Ducoffe</i>	43:1–43:16
On Computing the Average Distance for Some Chordal-Like Graphs <i>Guillaume Ducoffe</i>	44:1–44:16
A Cubic Vertex-Kernel for TRIVIAALLY PERFECT EDITING <i>Maël Dumas, Anthony Perez, and Ioan Todinca</i>	45:1–45:14
Lower Bounds on Avoiding Thresholds <i>Robert Ferens, Marek Szykula, and Vojtěch Vorel</i>	46:1–46:14
HyperLTL Satisfiability Is Σ_1^1 -Complete, HyperCTL* Satisfiability Is Σ_1^2 -Complete <i>Marie Fortin, Louwe B. Kuijjer, Patrick Totzke, and Martin Zimmermann</i>	47:1–47:19
Matching Patterns with Variables Under Hamming Distance <i>Paweł Gawrychowski, Florin Manea, and Stefan Siemer</i>	48:1–48:24

Keyboards as a New Model of Computation <i>Yoan Gérard, Bastien Laboureix, Corto Mascle, and Valentin D. Richard</i>	49:1–49:20
Quantum Speedups for Dynamic Programming on n -Dimensional Lattice Graphs <i>Adam Glos, Martins Kokainis, Ryuhei Mori, and Jevgēnijs Vihrovs</i>	50:1–50:23
A Note on the Join of Varieties of Monoids with LI <i>Nathan Grosshans</i>	51:1–51:16
Optimal Regular Expressions for Palindromes of Given Length <i>Hermann Gruber and Markus Holzer</i>	52:1–52:15
A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct <i>Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann</i>	53:1–53:20
Perfect Forests in Graphs and Their Extensions <i>Gregory Gutin and Anders Yeo</i>	54:1–54:13
On Deciding Linear Arithmetic Constraints Over p -adic Integers for All Primes <i>Christoph Haase and Alessio Mansutti</i>	55:1–55:20
Obstructing Classification via Projection <i>Pantea Haghighatkhah, Wouter Meulemans, Bettina Speckmann, Jérôme Urhausen, and Kevin Verbeek</i>	56:1–56:19
Online Domination: The Value of Getting to Know All Your Neighbors <i>Hovhannes A. Harutyunyan, Denis Pankratov, and Jesse Racicot</i>	57:1–57:21
A Linear-Time Nominal μ -Calculus with Name Allocation <i>Daniel Hausmann, Stefan Milius, and Lutz Schröder</i>	58:1–58:18
Test of Quantumness with Small-Depth Quantum Circuits <i>Shuichi Hirahara and François Le Gall</i>	59:1–59:15
On Search Complexity of Discrete Logarithm <i>Pavel Hubáček and Jan Václavěk</i>	60:1–60:16
A Homological Condition on Equational Unifiability <i>Mirai Ikebuchi</i>	61:1–61:16
Ordered Fragments of First-Order Logic <i>Reijo Jaakkola</i>	62:1–62:14
The Simplest Non-Regular Deterministic Context-Free Language <i>Petr Jančar and Jiří Šíma</i>	63:1–63:18
On the Hardness of Compressing Weights <i>Bart M. P. Jansen, Shivesh K. Roy, and Michał Włodarczyk</i>	64:1–64:21
Griddings of Permutations and Hardness of Pattern Matching <i>Vít Jelínek, Michal Opler, and Jakub Pekárek</i>	65:1–65:22
Sets of Linear Forms Which Are Hard to Compute <i>Michael Kaminski and Igor E. Shparlinski</i>	66:1–66:22
On Positivity and Minimality for Second-Order Holonomic Sequences <i>George Kenison, Oleksiy Klurman, Engel Lefaucheur, Florian Luca, Pieter Moree, Joël Ouaknine, Markus A. Whiteland, and James Worrell</i>	67:1–67:15

Improved Upper Bounds for the Rigidity of Kronecker Products <i>Bohdan Kivva</i>	68:1–68:18
The Power of One Clean Qubit in Communication Complexity <i>Hartmut Klauck and Debbie Lim</i>	69:1–69:23
Connecting Constructive Notions of Ordinals in Homotopy Type Theory <i>Nicolai Kraus, Fredrik Nordvall Forsberg, and Chuangjie Xu</i>	70:1–70:16
Maximum Votes Pareto-Efficient Allocations via Swaps on a Social Network <i>Fu Li and Xiong Zheng</i>	71:1–71:16
Finite Models for a Spatial Logic with Discrete and Topological Path Operators <i>Sven Linker, Fabio Papacchini, and Michele Sevegnani</i>	72:1–72:16
Recursive Backdoors for SAT <i>Nikolas Mählmann, Sebastian Siebertz, and Alexandre Vigny</i>	73:1–73:18
Parallel Algorithms for Power Circuits and the Word Problem of the Baumslag Group <i>Caroline Mattes and Armin Weiß</i>	74:1–74:24
The Complexity of Transitively Orienting Temporal Graphs <i>George B. Mertzios, Hendrik Molter, Malte Renken, Paul G. Spirakis, and Philipp Zschoche</i>	75:1–75:18
Temporal Reachability Minimization: Delaying vs. Deleting <i>Hendrik Molter, Malte Renken, and Philipp Zschoche</i>	76:1–76:15
A Timecop’s Chase Around the Table <i>Nils Morawietz and Petra Wolf</i>	77:1–77:18
Syntactic Minimization Of Nondeterministic Finite Automata <i>Robert S. R. Myers and Henning Urbat</i>	78:1–78:16
Idempotent Turing Machines <i>Keisuke Nakano</i>	79:1–79:18
Ergodic Theorems and Converses for PSPACE Functions <i>Satyadev Nandakumar and Subin Pulari</i>	80:1–80:19
On Guidable Index of Tree Automata <i>Damian Niwiński and Michał Skrzypczak</i>	81:1–81:14
Feedback Vertex Set and Even Cycle Transversal for H -Free Graphs: Finding Large Block Graphs <i>Giacomo Paesani, Daniël Paulusma, and Paweł Rzażewski</i>	82:1–82:14
Stabilization Bounds for Influence Propagation from a Random Initial State <i>Pál András Papp and Roger Wattenhofer</i>	83:1–83:15
Parameterized (Modular) Counting and Cayley Graph Expanders <i>Norbert Peyerimhoff, Marc Roth, Johannes Schmitt, Jakob Stix, and Alina Vdovina</i>	84:1–84:15
A Hierarchy of Nondeterminism <i>Bader Abu Radi, Orna Kupferman, and Ofer Leshkowitz</i>	85:1–85:21

0:x **Contents**

Boolean Automata and Atoms of Regular Languages <i>Hellis Tamm</i>	86:1–86:13
The Gödel Fibration <i>Davide Trotta, Matteo Spadetto, and Valeria de Paiva</i>	87:1–87:16
Abstract Congruence Criteria for Weak Bisimilarity <i>Stelios Tsampas, Christian Williams, Andreas Nuyts, Dominique Devriese, and Frank Piessens</i>	88:1–88:23
Quantum Multiple-Valued Decision Diagrams in Graphical Calculi <i>Renaud Vilmart</i>	89:1–89:15
Decision Problems for Origin-Close Top-Down Tree Transducers <i>Sarah Winter</i>	90:1–90:16

■ Preface

The International Symposium on Mathematical Foundations of Computer Science (MFCS conference series) is a well-established venue for presenting research results in theoretical computer science. The broad scope of the conference encourages interactions between researchers who might not meet at more specialized venues. The first MFCS conference was organized in 1972 in Jabłonna (near Warsaw, Poland). Since then, the conference traditionally moved between the Czech Republic, Slovakia, and Poland. More recently, the conference started traveling to other European countries, including Denmark, the United Kingdom, Germany. The venue for this – the 46th – edition of MFCS, is Tallinn, Estonia.

The program committee of MFCS 2021 accepted 85 papers out of 199 submissions, with the authors of the submitted papers representing over 35 countries. We would like to express our deep gratitude to all the committee members and reviewers for their extensive reports and discussions on the merits of the submissions. Due to the Covid-19 pandemic MFCS 2021 was held as a hybrid event. It featured invited talks by Amina Doumane (ENS Lyon), Martin Grohe (RWTH Aachen University), Joël Ouaknine (Max Planck Institute for Software Systems), Eva Rotenberg (Technical University of Denmark), and Barna Saha (UC Berkeley) on topics that reflected the broad scope of the conference.

MFCS proceedings have been published in the Dagstuhl/LIPIcs series since 2016. We would like to thank Michael Wagner and the LIPIcs team for all their kind help and support. We also warmly thank the organising committee of MFCS, chaired by Pawel Sobocinski, for their hard work in setting up and running the event.

Filippo Bonchi
Simon J. Puglisi



■ Conference Organization

Program Committee

Filippo Bonchi	University of Pisa, co-chair
Tiziana Calamoneri	Sapienza University of Rome
Corina Cirstea	University of Southampton
Laure Daviaud	City, University of London
Vida Dujmovic	University of Ottawa
Leah Epstein	University of Haifa
Henning Fernau	Universität Trier
Pawel Gawrychowski	University of Wrocław
Telikepalli Kavitha	Tata Institute of Fundamental Research
Stefan Kiefer	University of Oxford
Aleks Kissinger	University of Oxford
Christian Komusiewicz	Philipps-Universität Marburg
Lukasz Kowalik	University of Warsaw
Daniel Král	Masaryk University
Oded Lachish	Birkbeck, University of London
Sophie Laplante	IRIF, Université Paris Diderot Paris 7
Daniel Lokshtanov	UCSB
Giulio Manzonetto	Université Sorbonne Paris-Nord
Nicole Megow	Universität Bremen
Stefan Milius	FAU Erlangen-Nürnberg
Matteo Mio	CNRS and ENS Lyon
Koko Muroya	Kyoto University
Eunjin Oh	POSTECH
Yoshio Okamoto	The University of Electro-Communications
Neil Olver	London School of Economics and Political Science
Diana Pigué	Czech Academy of Sciences
Nadia Pisanti	University of Pisa
Simon J. Puglisi	University of Helsinki, co-chair
Rajeev Raman	University of Leicester
Francesco Ranzato	University of Padova
Peter Rossmanith	RWTH Aachen University
Jurriaan Rot	Radboud University
Laura Sanita	University of Waterloo
Shikha Singh	Williams College
Ana Sokolova	University of Salzburg
David Spivak II	Topos Institute
Tatiana Starikovskaya	École Normale Supérieure
Szymon Toruńczyk	University of Warsaw
Ryuhei Uehara	Japan Advanced Institute of Science and Technology
Prudence Wong	University of Liverpool
Fabio Zanasi	University College London
Meirav Zehavi	Ben-Gurion University
Valeria de Paiva	Topos Institute and PUC-RJ

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).
Editors: Filippo Bonchi and Simon J. Puglisi



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

External Reviewers

Sivert Aasnaess	Amir Abboud	Eric Allender
Shinwoo An	Eugene Asarin	Eric Bach
Miriam Backens	Patrick Baillot	Georgios Bakirtzis
Chris Barrett	Libor Barto	James Bartusek
Julien Baste	Matías R. Bender	Matthias Bentert
Petra Berenbrink	Jeremias Berg	Giulia Bernardini
Anna Bernasconi	Nick Bezhanishvili	Siddharth Bhaskar
Marcin Bienkowski	Vittorio Bilò	Eric Blais
Achim Blumensath	Markus Bläser	Udi Boker
Ilario Bonacina	Marcello Bonsangue	Kaustav Bose
Prosenjit Bose	Olivier Bournez	Nicolas Bousquet
Lukasz Bożyk	Julian Bradford	Joshua Brakensiek
Cyril Branciard	Tomasz Brengos	Gavin Brennen
Davide Bresolin	Marcin Briański	Ulrik Buchholtz
Andrei Bulatov	Martin Bullinger	Jakub Bulín
Elisabet Burjons	Laurine Bénéteau	Christopher Cade
Michaël Cadilhac	Miguel Calejo	Marco Carbone
Titouan Carlette	Lorenzo Carlucci	Antonio Casares
Davide Castelnovo	Jérémie Chalopin	Yi-Jun Chang
Giulio Chiribella	Dmitry Chistikov	Janka Chlebikova
Kyungjin Cho	Keerti Choudhary	Marek Chrobak
Valentina Ciriani	Ilan Cohen	Alessio Conte
Andrea Corradini	Federico Corò	Alfredo Costa
Ágnes Cseh	Peter Csikvari	Radu Curticapean
Mina Dalirrooyfard	Niel De Beaudrap	Gianluca De Marco
Mateus De Oliveira Oliveira	Marc de Visme	Dario Della Monica
Dariusz Dereniowski	Silvia Di Gregorio	Marco Di Summa
Volker Diekert	Francesco Dolce	Jinshuo Dong
Kyveli Doveri	Jan Dreier	Ran Duan
Eliana Duarte	Saska Dönges	Richard East
Eduard Eiben	Ahmed El Alaoui	Thomas Erlebach
Martin Escardo	Durand Fabien	Jittat Fakcharoenphol
Piotr Faliszewski	Jerome Feret	Nathanaël Fijalkow
Eldar Fischer	Dana Fisman	Mário Florido
Fedor Fomin	Dominik D. Freydenberger	Soichiro Fujii
Wesley Fussner	Moses Ganardi	Robert Ganian
Ziyuan Gao	Evangelia Gergatsouli	Sevag Gharibian
Dan Ghica	Stefano Gogioso	Massimiliano Goldwurm
Chaim Goodman-Strauss	Martin Grohe	Niels Grüttemeier
Zeyu Guo	Gregory Gutin	Amar Hadzihasanovic
Magnús M. Halldórsson	Yassine Hamoudi	Tero Harju
Tim A. Hartmann	Pavol Hell	Dylan Hendrickson
Juho Hirvonen	Duc A. Hoang	Stefan Hoffmann
Jana Hofmann	Markus Holzer	Rostislav Horcik
Tim Hosgood	Giovambattista Ianni	Rasmus Ibsen-Jensen
Pawel Idziak	Christian Ikenmeyer	Simon Iosti

Taisuke Izumi	Daniele Izzi	Bart Jacobs
Wojciech Janczewski	Matthew Jenssen	Mark Jones
Konstanty Junosza-Szaniawski	Frank Kammer	Yoav Kantor
Christos Kapoutsis	Jarkko Kari	George Kenison
Arindam Khan	Yihan Kim	Kei Kimura
Evangelos Kipouridis	Bartek Klin	Remke Kloosterman
Dušan Knop	Yasuaki Kobayashi	Mikko Koivisto
Christian Konrad	Maria Kosche	Dexter Kozen
Laszlo Kozma	Jan Kretinsky	Hlér Kristjánsson
Jacek Krzaczkowski	Mrinal Kumar	Michal Kunc
Denis Kuperberg	Greg Kuperberg	Martin Kurečka
Barbara König	Sébastien Labbé	Ander Lamaison
Massimo Lauria	Francois Le Gall	Thierry Lecroq
Karoliina Lehtinen	Aurélien Lemay	Giacomo Lenzi
Nathan Lhote	Bo Li	Yinan Li
Alexis Linard	Sven Linker	Markus Lohrey
Florian Lonsing	Fosco Loregian	Henri Lotze
Gabor Lugosi	Vladimir Lysikov	Christof Löding
Urmila Mahadev	Anil Maheshwari	Frederik Mallmann-Trenn
Sebastian Maneth	Matteo Manighetti	Pasin Manurangsi
Radu Mardare	Johannes Marti	Theo Mary
Tomas Masopust	Simon Mauras	Elvira Mayordomo
Samuel McCauley	Marc Mezzarobba	Mehdi Mhalla
Samuel Mimram	Neeldhara Misra	Dieter Mitsche
Yoshihiro Mizoguchi	Daniel Mock	Joshua Moerman
Samuel Mohr	Hendrik Molter	Nils Morawietz
Pat Morin	Larry Moss	Norbert Th. Müller
Torsten Mütze	Stefan Neumann	André Nichterlein
Joachim Niehren	Naomi Nishimura	Nicolas Nisse
Alexandre Nolin	Jan Obdrzalek	Alexander Okhotin
Sebastian Ordyniak	Jan Otop	Joel Ouaknine
Tommaso Padoan	Anurag Pandey	Francesco Parolini
Paweł Parys	Francesco Pasquale	Erik Paul
Daniel Paulusma	Kristyna Pekarkova	Guillermo Perez
Sylvain Perifel	Jeff Phillips	Marta Piecyk
Robin Piedeleu	Théo Pierron	Georgios Piliouras
Veronika Pillwein	Thomas Place	Karol Pokorski
Filip Pokrývka	Alberto Policriti	Federico Poloni
Andrew Polonsky	Aditya Potukuchi	Damien Pous
John Power	Pierre Pradic	Ian Pratt-Hartmann
Nicola Prezza	David Purser	Marco Túlio Quintino
Indhumathi Raman	Fariba Ranjbar	Carl Philipp Reh
Felix Reidl	Adele Rescigno	Pierre-Alain Reynier
Colin Riba	Cordian Riener	Maurice Rojas
Günter Rote	Bodhayan Roy	Subhayan Roy Moulik
Katarzyna Rybarczyk	Andrew Ryzhikov	Julian Sahasrabudhe
Ivano Salvo	Rudini Sampaio	Miklos Santha
Saket Saurabh	Joe Sawada	Alceste Scalas

Šimon Schierreich	Pascal Schweitzer	Lia Schütze
Helmut Seidl	Paolo Serafino	Jiří Sgall
Alexander Shen	Mahsa Shirmohammadi	Yaroslav Shitov
Sebastian Siebertz	Florian Sikora	Blerina Sinaimeri
Makrand Sinha	Michał Skrzypczak	Friedrich Slivovsky
Pawel Sobocinski	Frank Sommer	Gaurav Sood
Aikaterini Sotiraki	Matteo Spadetto	Jakob Spooner
Toby St Clere Smithe	Caleb Stanford	Howard Straubing
Georg Struth	Donald Stull	Ondrej Suchy
Anupa Sunny	Michelle Sweering	Tony Tan
Seiichiro Tani	Till Tantau	David Tench
Hélène Touzet	Lisa Tse	Sean Tull
Henning Urvat	Jouko Vaananen	Rohit Vaish
Wim van Dam	John van de Wetering	Martijn van Ee
Rob van Glabbeek	Gerco van Heerdt	Rob van Stee
Yann Vaxès	Thomas Vidick	Gilles Villard
Marc Vinyals	Sundar Vishwanathan	Ben Lee Volk
Mikhail Volkov	Magnus Wahlström	Tomasz Walen
Pascal Weil	Omri Weinstein	Armin Weiss
Ben Wiederhake	Sarah Winter	Thorsten Wißmann
Gerhard J. Woeginger	Petra Wolf	James Worrell
Mingyu Xiao	Yukiko Yamauchi	Tetsuo Yokoyama
Vladimir Zamdzhiev	Christina Zarb	Noam Zeilberger
Linpeng Zhang	Dmitriy Zhuk	Martin Zimmermann

Non-Axiomatizability of the Equational Theories of Positive Relation Algebras

Amina Doumane ✉

CNRS, ENS Lyon, France

Abstract

In the literature, there are two ways to show that the equational theory of relations over a given signature is not finitely axiomatizable. The first-one is based on games and a construction called Rainbow construction. This method is very technical but it shows a strong result: the equational theory cannot be axiomatized by any finite set of *first-order formulas*. There is another method, based on a graph characterization of the equational theory of relations, which is easier to get and to understand, but proves a weaker result: the equational theory cannot be axiomatized by any finite set of *equations*.

In this presentation, I will show how to complete the second technique to get the stronger result of non-axiomatizability by first-order formulas.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases Relation algebra, Graph homomorphism, Equational theories, First-order logic

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.1

Category Invited Talk



© Amina Doumane;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 1; pp. 1:1–1:1

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A Deep Dive into the Weisfeiler-Leman Algorithm

Martin Grohe  

RWTH Aachen University, Germany

Abstract

The Weisfeiler-Leman algorithm is a well-known combinatorial graph isomorphism test going back to work of Weisfeiler and Leman in the late 1960s. The algorithm has a surprising number of seemingly unrelated characterisations in terms of logic, algebra, linear and semi-definite programming, and graph homomorphisms. Due to its simplicity and efficiency, it is an important subroutine of all modern graph isomorphism tools. In recent years, further applications in linear optimisation, probabilistic inference, and machine learning have surfaced.

In my talk, I will introduce the Weisfeiler-Leman algorithm and some extensions. I will discuss its expressiveness and the various characterisations, and I will speak about its applications.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Logic

Keywords and phrases Weisfeiler-Leman algorithm, graph isomorphism, counting homomorphisms, finite variable logics

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.2

Category Invited Talk

References

- 1 Martin Grohe. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–16, 2020. doi:10.1145/3375395.3387641.
- 2 Martin Grohe. The logic of graph neural networks. In *Proceedings of the 36th ACM-IEEE Symposium on Logic in Computer Science*, 2021.
- 3 Martin Grohe, Kristian Kersting, Martin Mladenov, and Pascal Schweitzer. Color refinement and its applications. In G. Van den Broeck, K. Kersting, S. Natarajan, and D. Poole, editors, *An Introduction to Lifted Probabilistic Inference*. Cambridge University Press, 2021. To appear. URL: https://www.lics.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaabttcqu.
- 4 Martin Grohe, Pascal Schweitzer, and Daniel Wiebking. Deep Weisfeiler Leman. In *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms*, pages 2600–2614, 2021. doi:10.1137/1.9781611976465.154.
- 5 Sandra Kiefer. The Weisfeiler-Leman algorithm: An exploration of its power. *ACM SIGLOG News*, 7(3):5–27, 2020.
- 6 Christopher Morris, Matthias Fey, and Nils M. Kriege. The power of the Weisfeiler-Leman algorithm for machine learning with graphs. *ArXiv*, 2105.05911, 2021. arXiv:2105.05911.
- 7 B.Y. Weisfeiler and A.A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series 2*, 1968. English translation by G. Ryabov available at https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf.



© Martin Grohe;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 2; pp. 2:1–2:1

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Holonomic Techniques, Periods, and Decision Problems

Joël Ouaknine  

Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Holonomic techniques have deep roots going back to Wallis, Euler, and Gauss, and have evolved in modern times as an important subfield of computer algebra, thanks in large part to the work of Zeilberger and others over the past three decades (see, e.g., [3, 2]). In this talk, I give an overview of the area, and in particular present a select survey of known and original results on decision problems for holonomic sequences and functions. I also discuss some surprising connections to the theory of periods and exponential periods, which are classical objects of study in algebraic geometry and number theory; in particular, I relate the decidability of certain decision problems for holonomic sequences to deep conjectures about periods and exponential periods, notably those due to Kontsevich and Zagier.

Parts of this exposition draws upon [1].

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Holonomic and hypergeometric sequences, Inequality problems, Continued fractions, Periods

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.3

Category Invited Talk

Funding *Joël Ouaknine*: ERC grant AVS-ISS (648701), and DFG grant 389792660 as part of TRR 248 (see <https://perspicuous-computing.science>). Also affiliated with Keble College, Oxford as *emmy.network* Fellow.

References

- 1 George Kenison, Oliver Klurman, Engel Lefauchaux, Florian Luca, Pieter Moree, Joël Ouaknine, Markus A. Whiteland, and James Worrell. On positivity and minimality for second-order holonomic sequences. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021), Tallinn, Estonia*, volume 202 of *LIPICs*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.67.
- 2 Marko Petkovšek, Herbert Wilf, and Doron Zeilberger. *A=B*. A. K. Peters, 1997.
- 3 Doron Zeilberger. A holonomic systems approach to special functions identities. *Journal of Computational and Applied Mathematics*, 32(3):321–368, 1990.



© Joël Ouaknine;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 3; pp. 3:1–3:1

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

On Dynamic Graphs

Eva Rotenberg  

Technical University of Denmark, Lyngby, Denmark

Abstract

In graph algorithms, many questions about a graph can be answered in time proportional to the size of the input, and such *linear time algorithms* are considered the epitome of efficiency. However, when the graph changes slightly, e.g. by the insertion or deletion of an edge or a vertex, it is undesirable to consider the entire input again. Rather, one would wish to keep some of the partial answers to questions about the old graph, and re-use them when computing answers to questions about the resulting graph. The art of handling such changes is studied in *dynamic graph algorithms*.

In this talk, we will see some examples of ideas and techniques for efficiently maintaining knowledge about a dynamically changing graph. We will consider classical and natural graph properties such as connectivity and planarity, and we will focus on deterministic algorithms.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms

Keywords and phrases Graph algorithms, dynamic graphs, connectivity, planarity, matching, online algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.4

Category Invited Talk

Funding *Eva Rotenberg*: Independent Research Fund Denmark grants 2020-2023 (9131-00044B) “Dynamic Network Analysis” and 2018-2021 (8021-00249B), “AlgoGraph”, and the VILLUM Foundation grant 37507 “Efficient Recomputations for Changeful Problems”.



© Eva Rotenberg;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 4; pp. 4:1–4:1

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Sublinear Algorithms for Edit Distance

Barna Saha ✉

University of California at Berkeley, CA, USA

Abstract

The edit distance is a way of quantifying how similar two strings are to one another by counting the minimum number of character insertions, deletions, and substitutions required to transform one string into the other. A simple dynamic programming computes the edit distance between two strings of length n in $O(n^2)$ time, and a more sophisticated algorithm runs in time $O(n + t^2)$ where t is the distance (Landau, Myers and Schmidt, SICOMP 1998). In pursuit of obtaining faster running time, the last couple of decades have seen a flurry of research on approximating edit distance, including polylogarithmic approximation in near-linear time (Andoni, Krauthgamer and Onak, FOCS 2010), and a constant-factor approximation in subquadratic time (Chakrabarty, Das, Goldenberg, Koucký and Saks, FOCS 2018). In this talk, we will discuss recent progress that goes beyond linear time, and studies sublinear time algorithms for edit distance. We will also discuss the role preprocessing might play in designing fast algorithms.

This is a joint work with Elazar Goldenberg, Tomasz Kociumaka, Robert Krauthgamer, and Aviad Rubinfeld.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Edit distance, sublinear algorithms, string processing

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.5

Category Invited Talk



© B. Saha;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 5; pp. 5:1–5:1

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

An Approximation Algorithm for the Matrix Tree Multiplication Problem

Mahmoud Abo-Khamis ✉ 

RelationalAI, Berkeley, CA, USA

Ryan Curtin ✉ 

RelationalAI, Atlanta, GA, USA

Sungjin Im ✉


University of California, Merced, CA, USA

Benjamin Moseley ✉

Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA

Hung Ngo ✉

RelationalAI, Berkeley, CA, USA

Kirk Pruhs ✉ 

Department of Computer Science, University of Pittsburgh, PA, USA

Alireza Samadian ✉

Department of Computer Science, University of Pittsburgh, PA, USA

Abstract

We consider the Matrix Tree Multiplication problem. This problem is a generalization of the classic Matrix Chain Multiplication problem covered in the dynamic programming chapter of many introductory algorithms textbooks. An instance of the Matrix Tree Multiplication problem consists of a rooted tree with a matrix associated with each edge. The output is, for each leaf in the tree, the product of the matrices on the chain/path from the root to that leaf. Matrix multiplications that are shared between various chains need only be computed once, potentially being shared between different root to leaf chains. Algorithms are evaluated by the number of scalar multiplications performed. Our main result is a linear time algorithm for which the number of scalar multiplications performed is at most 15 times the optimal number of scalar multiplications.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Matrix Multiplication, Approximation Algorithm

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.6

Funding *Sungjin Im*: Supported in part by NSF grants CCF-1617653 and CCF-1844939.

Benjamin Moseley: Supported in part by a Google Research Award, an Infor Research Award, a Carnegie Bosch Junior Faculty Chair and NSF grants CCF-1824303, CCF-1845146, CCF-1733873 and CMMI-1938909.

Kirk Pruhs: Supported in part by NSF grants CCF-1907673, CCF-2036077 and an IBM Faculty Award.

Acknowledgements We want to thank David Fernandez-Baca for discussions and pointers related to Markovian phelogeny trees.

1 Introduction

An instance of the Matrix Tree Multiplication problem consists of an arborescence $T = (V, E)$. There is a positive integer dimension d_v associated with each vertex v , and a d_u by d_v matrix $M_{u,v}$ associated with each directed edge (u, v) . Let r be the root of T and L be the collection of leaves of T . The output is, for each leaf $\ell \in L$, the product of the matrices on the directed



© Mahmoud Abo-Khamis, Ryan Curtin, Sungjin Im, Benjamin Moseley, Hung Ngo, Kirk Pruhs, and Alireza Samadian;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 6; pp. 6:1–6:14



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

6:2 Matrix Tree Multiplication

path from the root r to that leaf ℓ (in that order). We restrict our attention to algorithms that use the standard matrix multiplication algorithm to multiply two matrices, which uses ijk scalar multiplications to multiply an i by j matrix by a j by k matrix. We evaluate algorithms based on the aggregate number of scalar multiplications that they use. If the tree T has a single leaf, then this is the classic Matrix Chain Multiplication problem that is commonly covered in the dynamic programming chapter of introductory algorithms textbooks (e.g. [4]). However, it is important to note that a Matrix Tree Multiplication instance is not equivalent to a disjoint collection of Matrix Chain Multiplication instances, one for each leaf. This is because multiplications that are shared between various chains need only be computed once, not once for each chain.

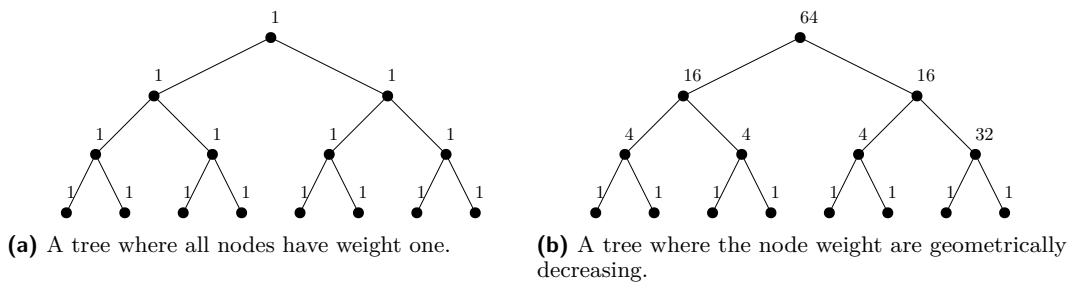
To help the reader appreciate this difference, let us consider two instances of Matrix Tree Multiplication where T is a balanced binary tree of depth $\lg n$ with n leaves. In the first instance, depicted in Figure 1a, all dimensions are 1. Then every feasible solution for every root-to-leaf path/chain uses $\lg n - 1$ scalar multiplications. However, the aggregate number of scalar multiplications can be quite different for different feasible solutions. To see this, if u is an ancestor of v in T , let $M_{u,v}$ denote the product of the matrices between u and v in T . We now consider two feasible solutions:

- **Top-Down:** For each root-to-leaf path $r = v_1, v_2, \dots, v_k$ in T , the i^{th} matrix multiplication is $M_{v_1, v_{i+2}} = M_{v_1, v_{i+1}} M_{v_{i+1}, v_{i+2}}$ for $i \in [1, k - 2]$.
- **Bottom-Up:** For each root-to-leaf path $r = v_1, v_2, \dots, v_k$ in T , the i^{th} matrix multiplication is $M_{v_{k-i-1}, v_k} = M_{v_{k-i-1}, v_{k-i}} M_{v_{k-i}, v_k}$ for $i \in [1, k - 2]$.

For Top-Down the computation of a matrix M_{v_1, v_h} can be shared by all root-to-leaf paths with leaves in the subtree T_{v_h} of T rooted at v_h . If we charge the computation of M_{v_1, v_h} to the vertex v_h , then each vertex that is neither the root nor a child of the root is charged exactly once and this charge is one. Thus, the objective value for Top-Down is $n - 3$. In contrast, for Bottom-Up none of its matrix multiplications can be shared between different paths. Thus, the objective value for the Bottom-Up algorithm is $\Theta(n \lg n)$. Thus, conceptually the advantage of Top-Down is that it minimizes the number of matrix multiplications, and maximizes the number of root-to-leaf paths that can utilize each particular matrix multiplication.

We remark that when the dimensions higher in the tree are significantly larger than the dimensions lower in the tree Bottom-Up can be significantly cheaper than Top-Down. This is because the individual matrix multiplications can be significantly cheaper. As an example consider the instance, depicted in Figure 1b. Here the dimension of a vertex at height h is 2^{2h} . Therefore, leaves have dimension 1, and the dimensions increase geometrically by a factor of 4 as one goes up the tree, with the root ultimately having dimension n^2 . The cost of Top-Down is clearly $\Omega(n^6)$ as there are individual matrix multiplications high in the tree that have this cost. On the other hand, the cost Bottom-Up is $O(n^5)$ as its cost increases geometrically up the tree, and the last matrix multiplications cost $O(n^5)$.

One motivation for our consideration of the Matrix Tree Multiplication problem comes from Markovian models of phylogenetic trees (see for example [9, Chapter 7]). In this setting the leaves are the taxa (for example, the DNA strands for known strains of some virus such as COVID), and the internal nodes represent conjectured historic ancestors of the leaf taxa. The phylogenetic tree T is thus a conjectured explanation of the evolutionary history of the leaf taxa. The matrices represent transition probabilities for mutation of a taxon in particular state to a taxon in some other state over some period of time. Multiplying the matrices on a root-to-leaf path results in an aggregate transition probability from an initial taxon state to a final leaf taxon state. This can then be used in a variety of ways, for example to find the



■ **Figure 1** Two example trees.

initial taxon state that most likely resulted in the leaf taxa states, or for determining the likelihood that T would result in the known leaf taxa given an initial distribution of states. Such applications motivate the consideration of the Matrix Tree Multiplication problem. Additionally, we are primarily interested because we believe that the problem is an interesting and natural generalization of a known classical algorithmic problem.

Another motivation for the Matrix Tree Multiplication problem comes from automatic differentiation (AD) [5, 1], which is widely used today in machine learning [1]. In AD, we are given a differentiable multivariate function $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$ for some p and q and our target is to compute the derivative of each of the q outputs with respect to each of the p inputs. Those derivatives can be arranged together in a $(q \times p)$ Jacobian matrix of f . The function f is typically represented as a computer program or a computation graph G which is a DAG where vertices and directed edges represent variables and elementary functions being applied to those variables. Now consider the special case where G has a tree structure $T = (V, E)$ where each vertex v corresponds to a set of d_v variables for some number d_v and each directed edge (u, v) corresponds to a multivariate function $f_{u,v} : \mathbb{R}^{d_u} \rightarrow \mathbb{R}^{d_v}$. Let $M_{u,v}$ be the transpose of the Jacobian matrix of $f_{u,v}$. Let r be the root of the tree and L the set of leaves. Thanks to the (multivariate) chain rule [5], the problem of computing the derivatives of the root's variables with respect to the variables of each leaf reduces to computing the multiplication of the matrices $M_{u,v}$ along each leaf-to-root path, hence to an instance of Matrix Tree Multiplication.

The standard textbook dynamic programming algorithm for the Matrix Chain Multiplication problem computes a parenthesization that results in the minimum number of scalar multiplications in time $O(n^3)$. This optimal parenthesization can be computed by a significantly more complicated algorithm that runs in time $O(n \log n)$ [7, 8]. It seems quite challenging to extend these approaches to the Matrix Tree Multiplication problem, even on very simple instances. For example, we do not know how to compute the optimal number of scalar multiplications in polynomial time even in the case that T has only 2 leaves. This is because it's not clear if there are subproblems the optimum solutions to which lead to that to the original problem.

Thus, we consider approximation algorithms. First, let us review what is known in terms of approximation algorithms for Matrix Chain Multiplication. [3] and [7] cite [2] as giving a 2-approximation algorithm for the Matrix Chain Multiplication problem.¹ A 1.25 approximation algorithm was later given in [3], and finally a 1.15 approximation algorithm was given in [6]. In each case an optimal parenthesization can be computed in linear or nearly linear time.

¹ [2] is an IBM technical report that does not seem to be available on the web, and the IBM library is closed during the COVID outbreak.

6:4 Matrix Tree Multiplication

Our main result is a linear-time 15-approximation algorithm, that we call the Cut Contraction algorithm, for the Matrix Tree Multiplication problem.

The rest of the paper is organized as follows. Section 2 gives a technical overview of the algorithm design and analysis. Section 3 introduces some additional notation and terminology that we will use. Section 4 describes the multiplications performed Cut Contraction algorithm (ignoring implementation details). Section 5 analyzes the approximation ratio for the Cut Contraction algorithm. Section 4.3 briefly discusses how to implement the Cut Contraction algorithm to get a linear-time algorithm that can output a parenthesization for each root to leaf path that has approximation ratio at most 15.

2 Technical Overview

To build some intuition, let us begin by giving a greedy 2-approximation algorithm for Matrix Chain Multiplication (which is presumably the algorithm given in [2]). Assume the vertices are $1, 2, \dots, n$. Let $m = \arg \min_i d_i$ be the index of the minimum dimension. If you think of the chain as a path graph, then d_m is the min vertex cut. Intuitively the algorithm multiplies the min-cut out to the end. So the algorithm first computes the matrix products $M_{m-i,m} = M_{m-i,m-i+1}M_{m-i+1,m}$ for $i = [2, m-1]$, $M_{m,m+i} = M_{m,m+i-1}M_{m+i-1,m+i}$ for $i = [2, n-m]$, and then finally computes $M_{1,n}$ by multiplying $M_{1,m}$ by $M_{m,n}$. This algorithm uses

$$d_1 d_n d_m + \sum_{i=1}^{m-2} d_i d_{i+1} d_m + \sum_{i=m+1}^{n-1} d_i d_{i+1} d_m$$

scalar multiplications. Observe that in any feasible solution it must be the case that for each $i \notin [m-1, m]$, the cost of the matrix multiplication that involves $M_{i,i+1}$ is at least $d_i d_{i+1} d_m$. Thus, a lower bound of the cost of optimal, that we call the edge cut lower bound, is:

$$\frac{d_1 d_n d_m + \sum_{i=1}^{m-2} d_i d_{i+1} d_m + \sum_{i=m+1}^{n-1} d_i d_{i+1} d_m}{2}$$

The factor of two comes from the fact that each matrix multiplication involves two matrices. Note that the upper bound on the cost of the algorithm is then twice this edge cut lower bound. Therefore, intuitively the edge cut lower bound assumes every edge/matrix gets multiplied by its preferred dimension, and this algorithm gives every edge its preference.

As a first step toward generalizing the algorithmic design to trees, we develop three cut based lower bounds for trees. The first lower bound is what we call the edge cut lower bound. Roughly speaking, the edge cut lower bound is

$$\sum_{(u,v) \in E} d_u d_v \alpha(u,v)/2$$

where $\alpha(u,v)$ is the minimum aggregate dimension of a cut that separates the edge (u,v) from either the root or the leaves. This edge cut lower bound assumes every edge/matrix gets multiplied by its preferred dimensions. The second lower bound is what we call the root-leaf lower bound. Roughly speaking, the root-leaf lower bound is

$$\sum_{\ell \in L} d_r d_\ell \beta(\ell)$$

where L is the set of leaves and $\beta(\ell)$ is the minimum dimension on the path from r to ℓ , excluding the endpoints. The final lower bound is what we call the vertex cut lower bound.

Roughly speaking, the vertex cut lower bound is

$$\sum_{v \in V} d_v \gamma(v)$$

where $\gamma(v)$ is the product of the min-cut separating v from the root and the aggregate dimensions of the min-cut separating v from the leaves.

With those cut lower bounds in hand, the natural path forward would be to design an algorithm in such a way that one could analyze its approximation ratio by comparing to these cut lower bounds. However, there are instances such that no matrix multiplication can be charged to the cut lower bounds (at least in a natural way). Further, there are instances where these cut lower bounds are too loose in aggregate and are more than a constant factor less than optimal. One such example is when T is a complete balanced binary tree where the dimension of the root is n , the dimension for vertices of height $h \in [0, \frac{\lg n}{2}]$ is 2^h , and the dimension of the rest of the vertices are \sqrt{n} . For this instance, all the cut lower bounds are $O(n^2)$, but the optimal solution has cost $\Theta(n^{5/2})$.

Our algorithm for Matrix Tree Multiplication first “reduces” the tree by performing all the matrix multiplications that can naturally be charged to the cut lower bounds. Roughly speaking the multiplications that can not be charged to these cut lower bounds are those in which the middle dimension d_v corresponds to a vertex v that is itself the min-cut of T_v , the subtree of T rooted at v . Thus, in the resulting reduced tree R , every node is the min-cut of its own subtree. Further it is relatively straight-forward to also ensure that the dimensions of the vertices on any root-to-leaf path in the reduced tree R form a geometrically decreasing sequence. Our algorithm then performs the multiplications on the reduced tree R in top-down order. We show that the above-mentioned properties of the reduced tree R are sufficient to allow us to use a charging argument to directly bound the cost of these top-down multiplications by a constant factor times the cost of any arbitrary feasible solution for T . Here we directly charge to the optimal and not the lower bounds.

3 Notation and Terminology

We use r to denote the root, and T_v to denote the subtree rooted at vertex v . For any vertex v and any sets of vertices A , let $\Pi_v(A)$ denote the set of vertices in A that are descendant of v . Given a set A , we denote the collective dimensions of the vertices in A by $W(A)$, that is $W(A) = \sum_{x \in A} d_x$.

We use $v \prec u$ to denote that v is a strict ancestor of u in T . We write \preceq to denote that v is an ancestor of u and also u could equal v . Given a vertex v of T , we call a collection C of vertices is a *cut* in T_v if the removal of the vertices in C leaves no remaining v to leaf path in T_v , and there are no two vertices u and v in C such that $v \prec u$. Given a vertex v of T , we call a cut C in T_v the *min-cut* of T_v if its vertices have the minimum cumulative dimensions among all cuts; that is $C = \arg \min_{C' \in \mathcal{C}} \sum_{v \in C'} d_v$.

4 The Cut Contraction Algorithm Description

Our Cut Contraction algorithm first partitions the tree T into various components. This is described in Subsection 4.1. An algorithm, which we call Reduce, then performs matrix multiplications that can be charged to the cut lower bounds. This is described in Subsection 4.2. Finally the Top-Down algorithm, described in the introduction, is applied to the resulting reduced tree R .

4.1 Classifying Vertices in the Tree

We explain how the vertices of T are classified by the algorithm before any matrix multiplications are performed. The first cut $C_1 = \mathcal{C}(r)$ is the min cut of T . To compute C_{i+1} , the algorithm iteratively considers the non-leaf vertices $u \in C_i$, and then considers all paths P from u to every leaf ℓ in T_u . Let v be the vertex with the least depth (closest to u) on P such that $d_v < d_u/2$. We call the vertex v a *checkpoint* vertex. If v does not exist then the leaf ℓ is included in C_{i+1} . If v exists then min-cut of the subtree T_v is added to C_{i+1} . Note that the min cut of T_v can be v itself.

Let D_i^v be the vertices between a non-leaf vertex $v \in C_i$ and the descendants of v in C_{i+1} including v and the descendants. That is, $D_i^v = \{u : v \preceq u \text{ and } \exists x \in C_{i+1} u \preceq x\}$. U_i^v are the vertices in D_i^v that prefer v over the cut C_{i+1} and S_i^v are the vertices that prefer C_{i+1} . Formally, $U_i^v = \{u : u \in D_i^v \text{ and } d_v \leq \sum_{w \in \Pi_u(C_{i+1})} d_w\}$ and $S_i^v = D_i^v \setminus U_i^v$. Finally $D_i = \cup_{v \in C_i} D_i^v$ are the level i intermediate vertices, $U_i = \cup_{v \in C_i} U_i^v$ are the level i upper vertices and $S_i = \cup_{v \in C_i} S_i^v$ are the level i lower vertices.

► **Observation 1.** For all vertices $v \in C_i$, the vertices in U_i^v is a connected component of T that includes v .

4.2 The Reduce Algorithm Description

Initially for every path $r = u_1, u_2, \dots, u_k$ from r to every vertex $u_k \in C_1$ of length at least two hops the algorithm computes the matrix products $M_{u_j, u_k} = M_{u_j, u_{j+1}} M_{u_{j+1}, u_k}$, for $j \in [1, k-2]$.

Next the algorithm iteratively performs matrix multiplication on matrices between C_i and C_{i+1} for $i = 1, 2, \dots$. To multiply matrices between C_i and C_{i+1} the algorithm iteratively considers vertices $v \in C_i$. The algorithm next iteratively considers vertices $u \in \Pi_v(C_{i+1})$. Let $v = u_1, u_2, \dots, u_k$ be the path from v to u . If $k \geq 3$ the algorithm then multiplies the matrices on this path in manner that we now describe (otherwise the algorithm does nothing on this path). Let m be minimum such u_{m+1} is not in U_i^v . Note that it could be that all of u_2, \dots, u_k are in S_i^v and thus $m = 1$, or all of u_1, \dots, u_k are U_i^v and thus $m = k$. If $m \geq 2$ the algorithm multiplies the matrices in U_i^v in top-down order. That is, it computes the matrix products $M_{v, u_j} = M_{v, u_{j-1}} M_{u_{j-1}, u_j}$ for $j \in [2, m]$. If $k - m \geq 2$ the algorithm multiplies the matrices in S_i^v in bottom-up order. That is, the algorithm computes the matrix products $M_{u_{k-j}, u_k} = M_{u_{k-j}, u_{k-j+1}} M_{u_{k-j+1}, u_k}$ for $j \in [2, k - m]$. Finally, if $2 \leq m \leq k - 1$ the algorithm computes the matrix product $M_{v, u_k} = M_{v, u_m} M_{u_m, u_k}$. Let the resulting tree be R .

4.3 Linear Time Implementation

Here we sketch the key steps to make the algorithm run in linear time. To find the min-cuts of *every* subtree, the algorithm can start from the leaves and make them the min-cut of their subtree. Recursively in a bottom up fashion the algorithm can find the minimum cut of all the subtrees. For each vertex, the algorithm compares its dimension with the summation of the min-cuts of its children.

Once this is known, C_1 can be found in linear time. In order to find the checkpoints and the next cuts, we only need to perform a depth first search over the tree. Similarly, finding the sets U_i^v and S_i^v can be done by a depth first search over the vertices of the tree. After this step, the multiplications are well defined.

5 Cut Contraction Approximation Analysis

In subsection 5.1 we state and prove three cut based lower bounds on optimal. In subsection 5.2 we prove some structural properties of the classification of vertices. In subsection 5.3 we analyze the Reduce algorithm. Finally in subsection 5.4 we analyze the Top-Down algorithm on the reduced tree R .

5.1 The Cut Lower Bounds

Let $\mathcal{C}(v)$ be a min-cut of the subtree T_v , and let $\mathcal{C}^-(v)$ be the minimum cut of T_v subject to the constraint that the cut does not contain v . Let $h(v)$ be the vertex x with the minimum dimension subject to the constraint that $r \preceq x \prec v$. For an edge $(u, v) \in E$ define $\alpha(u, v)$ as follows:

$$\alpha(u, v) = \begin{cases} d_{h(u)} & \text{if } u \neq r \text{ and } v \in L \\ W(\mathcal{C}^-(v)) & \text{if } u = r \text{ and } v \notin L \\ \min(d_{h(u)}, W(\mathcal{C}^-(v))) & \text{otherwise} \end{cases}$$

For a leaf $\ell \in L$ define $\beta(\ell)$ as follows:

$$\beta(\ell) = \min_{u \text{ s.t. } r \prec u \prec \ell} d_u$$

For a vertex $v \in V$ that is neither the root nor a leaf, define $\gamma(v)$ as follows

$$\gamma(v) = d_{h(v)} \cdot W(\mathcal{C}^-(v))$$

► **Lemma 2** (Edge Cut Lower Bound).

$$\sum_{(u,v) \in E} d_u d_v \alpha(u, v) \leq 2 \cdot \text{Opt}$$

Proof. Let $P_{u,v}$ be the set of all root-to-leaf paths passing an edge (u, v) . Let $A_{(u,v)}$ be the set of vertices q for which, the optimum algorithm has made a multiplication of cost $d_u d_v d_q$. That is, the algorithm has performed either the multiplication $M_{u,v} M_{v,q}$ or $M_{q,u} M_{u,v}$ and let $O_{u,v}$ be the total cost of these multiplications. That is, $O_{u,v} = \sum_{q \in A_{(u,v)}} d_u d_v d_q$.

In any feasible solution, for every path p in $P_{u,v}$, there should be one vertex in p that is in $A_{u,v}$. That is because, in order for the algorithm to find the final product of the matrices in p , at some point, it must multiply $M_{u,v}$ to some other matrix in p . Therefore, if no ancestor of u is in $A_{u,v}$, we know $A_{u,v}$ must be a cut (or its superset) in T_v that is not equal to $\{v\}$. If there exists a vertex $x \in A_{u,v}$ such that $x \prec u$, then we know $O_{u,v} \geq d_x d_u d_v \geq d_{h(u)} d_u d_v$. Otherwise, as $A_{u,v}$ is a cut in T_v , we have $W(A_{u,v}) \geq W(\mathcal{C}^-(v))$; thus, $O_{u,v} \geq W(A_{u,v}) d_u d_v \geq W(\mathcal{C}^-(v)) d_u d_v$. Therefore, in either case, $O_{u,v} \geq d_u d_v \alpha(u, v)$. Summing over all edges (u, v) , we get the following value for the total cost of the multiplications involving the matrix of an edge in T :

$$2\text{Opt} \geq \sum_{(u,v) \in E} O_{u,v} \geq \sum_{(u,v) \in E} d_u d_v \alpha(u, v)$$

We get the factor of 2 because each matrix multiplication only involves two matrices and therefore is counted at most twice. ◀

► **Lemma 3** (Root-Leaf Cut Lower Bound).

$$\sum_{\ell \in L} d_r d_\ell \beta(\ell) \leq \text{Opt}$$

Proof. Since the optimal solution is feasible, it must perform a multiplication of the form $M_{r,u}M_{u,\ell}$ for each leaf ℓ in order to compute $M_{r,\ell}$, which must cost at least $d_r d_\ell \beta(\ell)$, and cannot be shared among different leaves. ◀

► **Lemma 4** (Vertex Cut Lower Bound). *Let V' be the set of vertices in T that are neither a root nor a leaf. Then,*

$$\sum_{v \in V'} d_v \gamma(v) \leq \text{Opt}$$

Proof. Fix an edge (u, v) and consider all the root to leaf paths that pass through (u, v) . Any feasible solution needs to compute the final product of the matrices lying on all of these paths. We first prove the following claim: for any root-to-leaf path P that contains the edge (u, v) there exists a multiplication of the form $M_{a,u}M_{u,b}$ that the feasible solution computes where a and b are two vertices in P and $a \prec u \prec b$.

We can find this multiplication by the following procedure. We first consider the last multiplication that is performed on the path between r and a leaf node ℓ . Let $M_{r,w}M_{w,\ell}$, be that multiplication. If $w = u$, then we have found the multiplication. If $w \prec u$, then we recurse on the last multiplication that the algorithm has performed to calculate $M_{u,\ell}$ until we find a multiplication of the form $M_{a,u}M_{u,b}$. Lastly, if $u \prec w$, then we recurse on the last multiplication that the algorithm has performed to compute $M_{r,u}$.

Now, for an edge (u, v) , let $A_{(u,v)}$ be the set of all pairs of vertices (a, b) such that the algorithm has computed $M_{a,u}M_{u,b}$. From the above claim we can conclude that the set $B_{(u,v)} = \{b : (a, b) \in A_{(u,v)}\}$ is a cut in the subtree T_v , because for every root-to-leaf path that has the edge (u, v) , there exists a pair of (a, b) in $A_{(u,v)}$ that is on that path and $v \prec b$. Then, since these sets of multiplications are disjoint with respect to different edges (u, v) , we can get the following lower bound:

$$\text{Opt} \geq \sum_{(u,v) \in E} \sum_{(a,b) \in A_{(u,v)}} d_a d_b d_u \geq \sum_{(u,v) \in E} d_{h(u)} d_u \sum_{b \in B_{(u,v)}} d_b \geq \sum_{(u,v) \in E} d_{h(u)} d_u W(\mathcal{C}(v)).$$

Rewriting the last summation, by summing over all vertices u and then all the edges (u, v) connected to u , and the lemma follows:

$$\text{Opt} \geq \sum_{(u,v) \in E} d_{h(u)} d_u W(\mathcal{C}(v)) \geq \sum_{u \in V} d_u \gamma(u). \quad \blacktriangleleft$$

5.2 Structural Properties

Lemma 5 states that vertices between v and the cut $\mathcal{C}(v)$ inherit their min-cut from $\mathcal{C}(v)$. Lemma 6 lower bounds the size of min-cuts $\mathcal{C}(u)$ for vertices $u \in D_i$. Lemma 7 observes that the dimension of every vertex in a set C_i must be smaller than the dimension of any ancestor. Lemma 8 lower bounds the cut size for an edge $(u, w) \in D_i$. Lemma 9 observes that nodes in R are min-cuts of their subtree. Lemma 10 observes that the dimensions are geometrically decreasing on root to leaf paths in the reduced tree R .

► **Lemma 5.** *Let u be a descendant of v in T such that u also has a descendant in $\mathcal{C}(v)$. Then $\Pi_u(\mathcal{C}(v))$ is a min-cut in T_u .*

Proof. We prove the claim by contradiction. Let us assume $\Pi_u(\mathcal{C}(v))$ is not a min-cut of T_u . Note that there is no vertex $x \in \mathcal{C}(v)$ such that $x \preceq u$; because, if that was the case, we could remove any vertex in $\mathcal{C}(v)$ that is descendant of u and obtain a smaller cut. Therefore, since every v to leaf path including the ones passing through u have a vertex in $\mathcal{C}(v)$, we can conclude that $\Pi_u(\mathcal{C}(v))$ is a cut in T_u , and since we have assumed that it is not a min-cut, we can conclude $W(\mathcal{C}(u)) < W(\Pi_u(\mathcal{C}(v)))$.

Now we create a new cut in T_v by removing the vertices in $\Pi_u(\mathcal{C}(v))$ from $\mathcal{C}(v)$ and adding $\mathcal{C}(u)$. The weight of the new cut is $W(\mathcal{C}(v)) - W(\Pi_u(\mathcal{C}(v))) + W(\mathcal{C}(u))$ which is smaller than $W(\mathcal{C}(v))$ and that is a contradiction with the fact that $W(\mathcal{C}(v))$ was the min-cut of T_v . ◀

► **Lemma 6.** *For all nonleaf vertices $v \in C_i$, and for all vertices $u \in D_i^v$ it must be the case that $W(\mathcal{C}(u)) \geq \min(d_v/2, W(\Pi_u(C_{i+1})))$.*

Proof. If there is a vertex x in $\mathcal{C}(u)$ such that $d_x \geq d_v/2$ then the proof is trivial. Now we assume that for all vertices $x \in \mathcal{C}(u)$, we have $d_x < d_v/2$.

We divide the proof into two cases. In the first case assume that there is a checkpoint vertex t such that $v \prec t \preceq u$. Then by definition, $\Pi_t(C_{i+1})$ is a min-cut of T_t . Furthermore, since t is an ancestor of u , we have $\Pi_u(\Pi_t(C_{i+1})) = \Pi_u(C_{i+1})$. Then using Lemma 5, we can conclude $\Pi_u(C_{i+1})$ is a min-cut of T_u ; therefore, $W(\mathcal{C}(u)) = W(\Pi_u(C_{i+1}))$.

In the second case, assume that there is no checkpoint between v and u . Then for all the vertices $x \in \mathcal{C}(u)$, there exists a checkpoint vertex t such that $u \prec t \preceq x$; that is because $d_x \leq d_v/2$ and there is no checkpoint above v . Let T denote all such checkpoints, then T is a cut in T_u and a cut between u and $\mathcal{C}(u)$. Therefore, $\bigcup_{t \in T} \Pi_t(\mathcal{C}(u)) = \mathcal{C}(u)$, and based on the definition of C_{i+1} and the fact that T is a cut in T_u , we have $\Pi_u(C_{i+1}) = \bigcup_{t \in T} \Pi_t(C_{i+1}) = \bigcup_{t \in T} \mathcal{C}(t)$.

For any checkpoint in t , using Lemma 5, we know $\Pi_t(\mathcal{C}(u))$ is a min-cut of T_t , and as a result $W(\Pi_t(\mathcal{C}(u))) = W(\mathcal{C}(t)) = W(\Pi_t(C_{i+1}))$. Summing over all vertices in T we get $W(\mathcal{C}(u)) = W(\Pi_u(C_{i+1}))$. ◀

► **Lemma 7.** *For any nonleaf vertex $v \in C_i$ and for all ancestors u of v , it must be the case that $d_v \leq d_u$.*

Proof. We use induction on i . For the base case of $i = 1$, we know C_1 is the min-cut of T , and if there was a vertex u such that $u \prec v$ and $d_u < d_v$, we could create a smaller cut by replacing v with u in C_1 .

For $i > 1$, let q be the ancestor of v in C_{i-1} . We show that d_v is smaller than d_u for all vertices u where $q \preceq u \prec v$. Then by induction, it will be smaller than all of its ancestors because q is a non leaf vertex in C_{i-1} . Since v is not a leaf, there exists a checkpoint vertex t between v and q . Then as v is in $\mathcal{C}(t)$, for all u where $t \preceq u \preceq v$, we have $d_v \leq d_u$. Furthermore, based on the definition of a checkpoint, $d_t \leq d_q/2$ and for all vertices w where $q \preceq w \prec t$, we have $d_w > d_q/2$; therefore, $d_v \leq d_t \leq d_w$. ◀

► **Lemma 8.** *For all nonleaf vertices $v \in C_i$, and for all vertices edges (u, w) in T , with both endpoints in D_i^v we have $\alpha(u, w) \geq \min(d_v/2, W(\Pi_w(C_{i+1})))$.*

Proof. First, for every edge (u, w) with both ends in D_i^v , we prove

$$d_{h(u)} \geq \min(d_v/2, W(\Pi_w(C_{i+1}))). \quad (1)$$

6:10 Matrix Tree Multiplication

If there exists a checkpoint t such that $v \prec t \preceq u$, then for every vertex q such that $t \preceq q \preceq w$, we have $d_q \geq W(\mathcal{C}(w))$. This is because thanks to the definition of C_{i+1} and the fact that $q \preceq w$ we have $\Pi_t(C_{i+1}) = \mathcal{C}(t)$ and $\Pi_w(C_{i+1}) \subseteq \Pi_q(C_{i+1})$, and furthermore using Lemma 5, we know $\Pi_q(C_{i+1})$ is a min-cut of T_q and $\Pi_w(C_{i+1})$ is a min-cut of T_w . Therefore,

$$d_q \geq W(\mathcal{C}(q)) = W(\Pi_q(C_{i+1})) \geq W(\Pi_w(C_{i+1})) = W(\mathcal{C}(w)).$$

Moreover, for every vertex x where $v \preceq x \prec t$, we know $d_x \geq d_t$; therefore, we have $d_x \geq W(\mathcal{C}(w))$. Then, using Lemma 7, we can conclude $d_{h(u)} \geq W(\mathcal{C}(w)) = W(\Pi_w(C_{i+1}))$.

If there exists no checkpoint t between v and u , then based on the definition of a checkpoint and Lemma 7, we have $d_{h(u)} \geq d_v/2$. Thus, we have shown Eqn. (1).

Now, for every edge (u, w) with both ends in D_i^g , we prove

$$W(\mathcal{C}^-(w)) \geq \min(d_v/2, W(\Pi_u(C_{i+1}))). \quad (2)$$

First, note that $W(\mathcal{C}^-(w)) \geq W(\mathcal{C}(w))$ because $\mathcal{C}(w)$ is the minimum over all cuts including the cut $\{w\}$ whereas $\mathcal{C}^-(w) \neq \{w\}$. Furthermore, note that it is either the case that there is a checkpoint between v and every vertex in $\mathcal{C}(w)$ which implies $\mathcal{C}(w) = \Pi_w(C_{i+1})$, or $\mathcal{C}(w)$ has a vertex with dimension larger than $d_v/2$. Therefore, we have

$$W(\mathcal{C}^-(w)) \geq W(\mathcal{C}(w)) \geq \min(d_v/2, W(\Pi_w(C_{i+1}))).$$

Since $\alpha(u, w) = \min(d_{h(u)}, W(\mathcal{C}^-(w)))$, Eqn. (1) and (2) give the lemma. \blacktriangleleft

► Lemma 9. *Every vertex v in R that is not r , is the min-cut of both T_v and R_v .*

Proof. The fact that v is a min-cut of T_v follows from the definition of the cuts C_i and the definition of the Reduce algorithm. The fact that v is a min-cut of R_v follows from the fact that min-cuts of R_v are feasible cuts for T_v . \blacktriangleleft

► Lemma 10. *For every edge $(u, v) \in R$ such that $u \neq r$ and v is not a leaf, it must be the case that $d_u \geq 2d_v$.*

Proof. This is a direct consequence of the definition of the C_i 's. \blacktriangleleft

5.3 Reduce Analysis

► Lemma 11. *The cost incurred by the Reduce algorithm is at most $8 \cdot \text{Opt}$.*

Proof. We divide the multiplications into 4 categories and analyse their costs separately. We will refer to these costs as categories.

1. The multiplications that involve the matrices between the root and the vertices in C_1 .
2. The multiplications of the matrices with both ends in U_i^v for some $v \in C_i$.
3. The multiplications involving a matrix $M_{u,w}$ where (u, w) is an edge with w being in $S_i^v \cup \Pi_v(C_{i+1})$ for some $v \in C_i$.
4. The multiplications of the form $M_{v,m}M_{m,u}$ where $m \in U_i^v$ and $u \in \Pi_v(C_{i+1})$.

Note that the above categories cover all the multiplications done by the Reduce algorithm. We use the lower bound in Lemma 2, and show that the cost of each multiplication in the first three categories is a constant factor of $d_u d_v \alpha(u, v)$ for some edge (u, v) and no edge is charged more than once. Then we use the lower bound in Lemma 4 to bound the cost of the multiplications in the fourth category.

Category 1: Every matrix multiplication in this category has the form $M_{p(u)u} \cdot M_{u,v}$ where v is a vertex in C_1 and $M_{u,v}$ is the result of the product of the matrices in path between u and v for some vertex $v \in C_1$. Therefore, the cost of all multiplications in this category is

$$\sum_{(u,v) \in E_1} \sum_{x \in \Pi_v(C_1)} d_u d_v d_x,$$

where E_1 is the set of edges between the root and C_1 . This is because any matrix $M_{u,v}$ will be in one multiplication per each vertex of $\Pi_v(C_1)$. Note that since C_1 is a min-cut, for any subset B of C_1 , the value $\sum_{x \in B} d_x$ is smaller than the dimension of any of their common ancestors (otherwise, we could have got a smaller min-cut by replacing them with that ancestor). Therefore,

$$\sum_{(u,v) \in E_1} \sum_{x \in \Pi_v(C_1)} d_u d_v d_x = \sum_{(u,v) \in E_1} d_u d_v \alpha(u, v)$$

Category 2: Fix an integer i and a vertex $v \in C_i$. For every edge (u, w) with both ends in $U_i^v \setminus \{v\}$, the algorithm performs one multiplication of form $M_{v,u} M_{u,w}$ in top-down multiplication of U_i^v , and the cost for this multiplication is $d_v d_u d_w$. Using Lemma 8 and the definition of U_i^v , we know $\alpha(u, w) \geq \min(d_v/2, W(\Pi_w(C_{i+1}))) = d_v/2$. As a result the total cost of the multiplications in this category is bounded by

$$\sum_i \sum_{v \in C_i} \sum_{(u,w) \in E(U_i^v)} d_v d_u d_w \leq \sum_i \sum_{v \in C_i} \sum_{(u,w) \in E_2(v)} d_u d_w \alpha(u, w),$$

where $E_2(v)$ is the set of edges with both ends in U_i^v .

Category 3: Let u be a vertex in C_{i+1} and v be its ancestor in C_i . Then the path between u and v can be divided into two sections such that the vertices of the upper section are all in $U_i^v \cup \{v\}$ and the vertices of the lower section are in $S_i^v \cup \{u\}$. Then on the path between u and v , for every edge (w, t) on this path for which t is in $S_i^v \setminus \{u\}$, the algorithm performs the multiplication of form $M_{w,t} M_{t,u}$. Then if we sum over different vertices $u \in \Pi_t(C_{i+1})$, the total cost of the multiplications in this category that involve $M_{w,t}$ is $d_w d_t W(\Pi_t(C_{i+1}))$. Using the Lemma 8 and the definition of S_i^v , we have

$$2\alpha(w, t) \geq \min(d_v, 2W(\Pi_t(C_{i+1}))) \geq W(\Pi_t(C_{i+1})).$$

Therefore, the total cost of the multiplications in this category can be bounded by

$$\sum_i \sum_{v \in C_i} \sum_{(w,t) \in E_3(v)} d_w d_t W(\Pi_t(C_{i+1})) \leq \sum_i \sum_{v \in C_i} \sum_{(w,t) \in E_3(v)} 2d_w d_t \alpha(w, t),$$

where $E_3(v)$ is the set of edges (w, t) where t is in S_i^v .

Since the edges that are above C_1 , the edges that have both ends in $\bigcup_i \bigcup_{v \in C_i} U_i^v$, and the edges (u, w) with w being in $\bigcup_i \bigcup_{v \in C_i} S_i^v$ are disjoint, we have not double charged any edge. Therefore, using Lemma 2 we can conclude that the total cost of the multiplications in categories 1, 2, and 3 is at most 4Opt .

Category 4: Let u be a vertex in C_{i+1} and v be its ancestor in C_i . Let (q, w) be an edge on the path between v and u such that $q \in U_i^v$ and $w \in S_i^v$. The algorithm may make one multiplication of the form $M_{v,q} M_{q,u}$ for this path. Therefore, summing over all vertices $u \in \Pi_v(C_{i+1})$, we can derive the following total cost of all multiplications of this form:

6:12 Matrix Tree Multiplication

$$\sum_i \sum_{v \in C_i} \sum_{(q,w) \in E_4(v)} d_v d_q W(\Pi_w(C_{i+1}))$$

in which $E_4(v)$ is the set of edges with one end in U_v^i and one end in S_i^v .

For any edge $(q, w) \in E_4(v)$, using Lemma 6 and the fact that $w \in S_i^v$, we have

$$2W(\mathcal{C}(w)) \geq \min(d_v, 2W(\Pi_w(C_{i+1}))) \geq W(\Pi_w(C_{i+1})).$$

Furthermore, we know no checkpoint can be in U_v^i , because for any checkpoint t we have

$$W(\Pi_t(C_{i+1})) = W(\mathcal{C}(t)) \leq d_t \leq d_v/2.$$

Therefore, using Lemma 7 and the fact that no checkpoint is in U_v^i , for any edge $(q, w) \in E_4(v)$ we have $h(q) \geq d_v/2$, and we get the following upperbound on the cost of the multiplications in this category:

$$\sum_i \sum_{v \in C_i} \sum_{(q,w) \in E_4(v)} d_v d_q W(\Pi_w(C_{i+1})) \leq \sum_i \sum_{v \in C_i} \sum_{(q,w) \in E_4(v)} 4d_q d_{h(q)} W(\mathcal{C}(w)).$$

Then using Lemma 4, and taking the summation over all the edges that are not connected to the root we will get:

$$\sum_i \sum_{v \in C_i} \sum_{(q,w) \in E_4(v)} 4d_q d_{h(q)} W(\mathcal{C}(w)) \leq 4 \sum_{u \in V} d_u \gamma(u) \leq 4\text{Opt}. \quad \blacktriangleleft$$

5.4 Top-Down Analysis

Our analysis of the Top-Down algorithm on the reduced tree R is based on a charging argument. A few of the Top-Down multiplications will be charged to the root-leaf cut lower bound. However, most of the Top-Down matrix multiplications will be directly charged to various matrix multiplications in Opt. There are three different possible ways that the Top-Down matrix multiplications can be charged: leaf-charge, low-charge, and high-charge.

The charging is done independently for each root-to-leaf path P . Iteratively consider a fixed root-to-leaf path P in T ending in a leaf $\ell \in L$. Let $M_{r,u}M_{u,v}$ be a Top-Down matrix multiplication on P that has not yet leaf-charged or low-charged any multiplication in optimal. If there is no checkpoint between u and v in T , note that we must have $v = \ell$ and the root-leaf cut lower bound is charged. We call this a leaf-charge. Otherwise, assume $u \in C_i$, $v \in C_{i+1}$ and note that there must exist a checkpoint t strictly between u and v on P (and thus t can not be either the root r nor a child of the root r in T). Let $M_{r,a}M_{a,b}$ be an arbitrary matrix multiplication in the optimal solution such that $r \prec a \prec t \preceq b \preceq \ell$. We will show such a matrix multiplication must exist in the optimal solution in Lemma 12. If $b \preceq v$ then the optimal multiplication $M_{r,a}M_{a,b}$ is charged $d_r d_u d_v$, the cost of this Top-Down multiplication. Call this a low-charge. If $v \prec b$ then the optimal multiplication $M_{r,a}M_{a,b}$ is charged $d_r d_u d_b$, which is a fraction of the cost of this Top-Down multiplication. Call this a high-charge.

► **Lemma 12.** *For each root-to-leaf path P in T and for each vertex t on P that is neither the root nor a child of the root, at least one multiplication $M_{r,a}M_{a,b}$ is in the optimal solution for T such that $r \prec a \prec t \preceq b \preceq \ell$ where ℓ is the leaf in P .*

Proof. Let $x_1 \prec \dots \prec x_k = \ell$ be all the vertices in P such that the optimal solution contains a multiplication of the form $M_{r,x_i}M_{x_i,x_{i+1}}$. Note that it must be the case $k \geq 2$ since the optimal solution is feasible and it needs to compute M_{r,x_k} . The claim is there exists $j \in [1, k-1]$ such that $x_j \preceq t \preceq x_{j+1}$, and one can take $a = x_j$ and $b = x_{j+1}$.

To prove this claim, note that x_1 must be a child of root. This is because, otherwise, the optimal solution needs to perform another multiplication to compute M_{r,x_1} since it uses it in $M_{r,x_1}M_{x_1,x_2}$. Let $M_{r,x_0}M_{x_0,x_1}$ be that multiplication, then x_0 is on P which contradicts with the definition of x_1, \dots, x_k . Therefore, since t is not the root or its children, we have $x_1 \prec t$. Also we know x_k is the leaf and $t \preceq x_k$. Therefore, there exists a $j \in [1, k-1]$ such that $x_j \preceq t \preceq x_{j+1}$. ◀

► **Lemma 13.** *The aggregate amount of root-leaf cut charges is at most twice the root-leaf cut lower bound, and therefore at most $2 \cdot \text{Opt}$.*

Proof. For each leaf ℓ there can be at most one matrix multiplication, say $M_{r,u}M_{u,\ell}$ charged to it. From Lemma 7 and the fact that there is no check point between u and ℓ one can conclude that $\beta(\ell) \leq 2d_u$. ◀

► **Lemma 14.** *Every Top-Down matrix multiplication $M = M_{r,u}M_{u,v}$ charges at least $d_r d_u d_v$ to the multiplications in the optimal solution.*

Proof. If M was charged via a high charge, this is obvious. Otherwise assume M was only charged via low charges. Let $(a_1, b_1), \dots, (a_k, b_k)$ be the collection of multiplications in optimal that M was charged to via low charges. By the feasibility of the optimal solution $\{b_1, \dots, b_k\}$ must be a cut of T_v . Thus by Lemma 9 it must be the case that $\sum_{i=1}^k d_{b_i} \geq d_v$. Thus the aggregate amount of low charges is at least $d_r d_u d_v$. ◀

► **Lemma 15.** *Every matrix multiplication $M_{r,a}M_{a,b}$ in optimal is charged at most $2d_r d_a d_b$ by low charges.*

Proof. Let $M_{r,u}M_{u,v}$ with $u \in C_i$ and $v \in C_{i+1}$ be one of the multiplications of top-down that low-charges $M_{r,a}M_{a,b}$. Then we know there exists a checkpoint t in T such that $u \prec t \preceq v$ and $a \prec t \preceq b \preceq v$. Then the claim is that the only multiplications of top-down that may low-charge the multiplication $M_{r,a}M_{a,b}$ in optimal are the ones of the form $M_{r,u}M_{u,w}$ where $w \in \Pi_b(C_{i+1})$.

To see the reason for the above claim, consider any multiplication $M_{r,p}M_{p,q}$ that can low-charge $M_{r,a}M_{a,b}$. Based on the definition of low-charge, we have $p \in C_j$ is an ancestor of b , and $w \in C_{j+1}$ such that $b \preceq w$. If $j+1 \leq i$, then no vertex in C_{j+1} can be a descendent of u , and therefore, we cannot have $b \preceq w$ because that would imply $u \prec b \preceq w$. Furthermore, we cannot have $j \geq i+1$ because we already know that v is in C_{i+1} and $b \preceq v$; therefore, no vertex in C_j can be an ancestor of b , meaning we cannot have $p \prec b$; because that would mean $p \prec u$. Thus, the only possibility is $j = i$, which means $p = u$. Then the only vertices in C_{i+1} that are descendent of b are by definition $\Pi_b(C_{i+1})$.

Using this claim, we can conclude the maximum amount low-charged to a multiplication $M_{r,a}M_{a,b}$ is $d_r d_u W(\Pi_b(C_{i+1}))$. Note that $a \prec t$, therefore, $d_a > d_u/2$; that is because, all the vertices between t and u have dimensions larger than $d_u/2$, and using Lemma 7, we know the dimension of all the ancestors of u is at least d_u . Furthermore, note that $\Pi_t(C_{i+1})$ is the min-cut of T_t ; therefore, using lemma 5 and the fact that $t \preceq b \preceq v$, we can conclude $W(\Pi_b(C_{i+1})) < d_b$. Therefore, we can get the following upperbound for the total cost lower-charged to a multiplication $M_{r,a}M_{a,b}$ of optimal: $d_r d_u W(\Pi_b(C_{i+1})) \leq 2d_r d_a d_b$. ◀

► **Lemma 16.** *Every matrix multiplication $M_{r,a}M_{a,b}$ in optimal is charged at most $4d_r d_a d_b$ by high-charges.*

Proof. Consider a multiplication $M_{r,u}M_{u,v}$ in top-down that high-charges the multiplication $M_{r,a}M_{a,b}$ in optimal. We have $a \prec t \preceq v \preceq b$ where t is the checkpoint between u and v , and $M_{r,u}M_{u,v}$ high-charges the cost $d_r d_u d_b$. Note that both t and v are on the path between a and b in T . Let $(u_1, u_2), (u_2, u_3), \dots, (u_{k-1}, u_k)$ be the edges in R , for which u_2, \dots, u_k are all on the path between a and b in T , and the checkpoint between u_i, u_{i+1} for all i is also on this path. Using the definition of high-charge, the only multiplications in Top-Down that can high charge $M_{r,a}M_{a,b}$ are the multiplications of the form $M_{r,u_i}M_{u_i,u_{i+1}}$ for $i \in [1, k-1]$. Therefore, the total cost high-charged to $M_{r,a}M_{a,b}$ is at most: $d_r d_b \sum_{i=1}^k d_{u_i}$. Using Lemma 10, we know $d_{u_{i+1}} \leq d_{u_i}/2$. Furthermore, using the definition of the checkpoint and Lemma 7, we have $d_{u_1} \leq 2d_a$ because a is the ancestor of the checkpoint between u_1 and u_2 in T . Therefore, the total cost can be upper bounded as follows: $d_r d_b \sum_{i=1}^k d_{u_i} \leq d_r d_b d_{u_1} \sum_{i=0}^{\infty} 1/2^i \leq 4d_r d_b d_a$ ◀

We now can prove the main theorem.

► **Theorem 17.** *The Cut Contraction Algorithm for the Tree Matrix Multiplication problem is 15 approximate.*

Proof. Using Lemma 11, we can conclude the cost of Reduce multiplications is 8Opt, and using Lemmas 13, 14, 15, and 16, we can conclude the cost of the multiplications performed in TopDown phase is 7Opt which gives us total cost of 15Opt. ◀

6 Conclusions

In this paper we studied a natural extension of the matrix chain problem where multiples chains are overlaid forming a tree. Currently, we do not know if the problem is NP-hard although we believe so. The obvious open question is to show that the problem is indeed NP-hard. Further, we do not know how to obtain a better approximation using any polynomial time algorithms. Improving the approximation ratio would be another interesting direction. Finally, it would be very interesting to study the more general setting where the chains form an arbitrary DAG. The main challenge in such an extension seems to lie in discovering lower bounds different from what we used in this paper.

References

- 1 Atılım Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, 2017.
- 2 A. K. Chandra. Computing matrix chain products in near optimal time. *IBM Research Report*, RC 5625(24393), 1975. IBM T.J. Watson Research Center.
- 3 Francis Y. L. Chin. An $o(n)$ algorithm for determining a near-optimal computation order of matrix chain products. *Communications of the ACM*, 21(7):544–549, 1978.
- 4 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.
- 5 Andreas Griewank and Andrea Walther. *Evaluating Derivatives*. Society for Industrial and Applied Mathematics, second edition, 2008. doi:10.1137/1.9780898717761.
- 6 T. C. Hu and M. T. Shing. An $o(n)$ algorithm to find a near-optimum partition of a convex polygon. *Journal of Algorithms*, 2(2):122–138, 1981.
- 7 T. C. Hu and M. T. Shing. Computation of matrix chain products. part I. *SIAM Journal of Computing*, 11(2):362–373, 1982.
- 8 T. C. Hu and M. T. Shing. Computation of matrix chain products. part II. *SIAM Journal of Computing*, 13(2):228–251, 1984.
- 9 Mike Steel. *Phylogeny: Discrete and Random Processes in Evolution*. SIAM-Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2016.

Depth-First Search in Directed Planar Graphs, Revisited

Eric Allender   

Rutgers University, Piscataway, NJ, USA

Archit Chauhan  

Chennai Mathematical Institute, India

Samir Datta  

Chennai Mathematical Institute, India

Abstract

We present an algorithm for constructing a depth-first search tree in planar digraphs; the algorithm can be implemented in the complexity class $AC^1(UL \cap co-UL)$, which is contained in AC^2 . Prior to this (for more than a quarter-century), the fastest uniform deterministic parallel algorithm for this problem was $O(\log^{10} n)$ (corresponding to the complexity class $AC^{10} \subseteq NC^{11}$).

We also consider the problem of computing depth-first search trees in other classes of graphs, and obtain additional new upper bounds.

2012 ACM Subject Classification Theory of computation \rightarrow Complexity classes; Theory of computation \rightarrow Parallel algorithms

Keywords and phrases Depth-First Search, Planar Digraphs, Parallel Algorithms, Space-Bounded Complexity Classes

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.7

Related Version *Full Version*: <https://eccc.weizmann.ac.il/report/2020/074/>

Funding *Eric Allender*: Supported in part by NSF Grant CCF-1909216 and CCF-1909683.

Archit Chauhan: Partially supported by a grant from Infosys foundation and TCS PhD fellowship.

Samir Datta: Partially supported by a grant from Infosys foundation and SERB-MATRICES grant MTR/2017/000480.

1 Introduction

Depth-first search trees (DFS trees) constitute one of the most useful items in the algorithm designer's toolkit, and for this reason they are a standard part of the undergraduate algorithmic curriculum around the world. When attention shifted to parallel algorithms in the 1980's, the question arose of whether NC algorithms for DFS trees exist. An early negative result was that the problem of constructing the *lexicographically least* DFS tree in a given digraph is complete for P [20]. But soon thereafter significant advances were made in developing parallel algorithms for DFS trees, culminating in the RNC^7 algorithm of Aggarwal, Anderson, and Kao [1]. This remains the fastest parallel algorithm for the problem of constructing DFS trees in general graphs, in the probabilistic setting, or in the setting of nonuniform circuit complexity. It remains unknown if this problem lies in (deterministic) NC (and we do not solve that problem here).

More is known for various restricted classes of graphs. For directed acyclic graphs (DAGs), the lexicographically-least DFS tree from a given vertex can be computed in AC^1 [10]. (See also [11, 7, 13, 19, 16, 15].) For undirected planar graphs, an AC^1 algorithm for DFS trees was presented by Hagerup [14]. For more general planar directed graphs Kao and Klein presented an AC^{10} algorithm. Kao subsequently presented an AC^5 algorithm for DFS in *strongly connected* planar digraphs. In stating the complexity results for this prior work



© Eric Allender, Archit Chauhan, and Samir Datta;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 7; pp. 7:1–7:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in terms of complexity classes (such as AC^1, AC^{10} , etc.), we are ignoring an aspect that was of particular interest to the authors of this earlier work: minimizing the number of processors. This is because our focus is on classifying the complexity of constructing DFS trees in terms of complexity classes. Thus, if we reduce the complexity of a problem from AC^{10} to AC^2 , then we view this as a significant advance, even if the AC^2 algorithm uses many more processors (so long as the number of processors remains bounded by a polynomial). Indeed, our algorithms rely on the logspace algorithm for undirected reachability [21], which does not directly translate into a processor-efficient algorithm. We suspect that our approach can be modified to yield a more processor-efficient AC^3 algorithm, but we leave that for others to investigate.

1.1 Our Contributions

First, we observe that, given a DAG G , computation of a DFS tree in G logspace reduces to the problem of reachability in G . Thus, for general DAGs, computation of a DFS tree lies in NL, and for planar DAGs, the problem lies in $UL \cap \text{co-UL}$ [8, 23]. For classes of graphs where the reachability problem lies in L, so does the computation of DFS trees. One such class of graphs is planar DAGs with a single source (see [2], where this class of graphs is called SMPDs, for **S**ingle-source, **M**ultiple-sink, **P**lanar **D**AGs).

For undirected planar graphs, it was shown in [4] that the approach of Hagerup’s AC^1 DFS algorithm [14] can be adapted in order to show that construction of a DFS tree in a planar *undirected* graph logspace-reduces to computing the distance between two nodes in a planar digraph. Since this latter problem lies in $UL \cap \text{co-UL}$ [24], so does the problem of DFS for planar *undirected* graphs.

Our main contribution in the current paper is to show that a more sophisticated application of the ideas in [14] leads to an $AC^1(UL \cap \text{co-UL})$ algorithm for construction of DFS trees in planar *directed* graphs. (That is, we show DFS trees can be constructed by unbounded fan-in log-depth circuits that have oracle gates for a set in $UL \cap \text{co-UL}$.¹) Since $UL \subseteq NL \subseteq SAC^1 \subseteq AC^1$, the $AC^1(UL \cap \text{co-UL})$ algorithm can be implemented in AC^2 . Thus this is a significant improvement over the best previous parallel algorithm for this problem: the AC^{10} algorithm of [18], which has stood for 28 years.

2 Preliminaries

We assume that the reader is familiar with depth-first search trees (DFS trees).

We further assume that the reader is familiar with the standard complexity classes L, NL and P (see e.g. the text [6]). We will also make frequent reference to the logspace-uniform circuit complexity classes NC^k and AC^k . NC^k is the class of problems for which there is a logspace-uniform family of circuits $\{C_n\}$ consisting of AND, OR, and NOT gates, where the AND and OR gates have fan-in two and each circuit C_n has depth $O(\log^k n)$. (The logspace-uniformity condition implies that each C_n has only $n^{O(1)}$ gates.) AC^k is defined similarly, although the AND and OR gates are allowed unbounded fan-in. An equivalent characterization of AC^k is in terms of concurrent-read concurrent-write PRAMs with running time $O(\log^k n)$, using $n^{O(1)}$ processors. For more background on these circuit complexity classes, see, e.g., the text [26].

¹ An earlier version of this work claimed a stronger upper bound, but there was an error in one of the lemmas in that version [3].

A nondeterministic Turing machine is said to be *unambiguous* if, on every input x , there is at most one accepting computation path. If we consider logspace-bounded nondeterministic Turing machines, then unambiguous machines yield the class UL . A set A is in co-UL if and only if its complement lies in UL .

The construction of DFS trees is most naturally viewed as a *function* that takes a graph G and a vertex v as input, and produces as output an encoding of a DFS tree in G rooted at v . But the complexity classes mentioned above are all defined as sets of *languages*, instead of as sets of *functions*. Since our goal is to place DFS tree construction into the appropriate complexity classes, it is necessary to discuss how the complexity of functions fits into the framework of complexity classes.

When \mathcal{C} is one of $\{\text{L}, \text{P}\}$, it is fairly obvious what is meant by “ f is computable in \mathcal{C} ”; the classes of logspace-computable functions and polynomial-time-computable functions should be familiar to the reader. However, the reader might be less clear as to what is meant by “ f is computable in NL ”. As it turns out, essentially all of the reasonable possibilities are equivalent. Let us denote by FNL the class of functions that are computable in NL ; it is shown in [17] each of the three following conditions is equivalent to “ $f \in \text{FNL}$ ”.

1. f is computed by a logspace machine with an oracle from NL .
2. f is computed by a logspace-uniform NC^1 circuit family with oracle gates for a language in NL .
3. $f(x)$ has length bounded by a polynomial in $|x|$, and the set $\{(x, i, b) : \text{the } i^{\text{th}} \text{ bit of } f(x) \text{ is } b\}$ is in NL .

Rather than use the unfamiliar notation “ FNL ”, we will abuse notation slightly and refer to certain functions as being “computable in NL ”.

The proof of the equivalence above relies on the fact that NL is closed under complement. Thus it is far less clear what it should mean to say that a function is “computable in UL ” since it remains an open question if UL is closed under complement (although it is widely conjectured that $\text{UL} = \text{NL}$) [22, 5]). However the proof from [17] carries over immediately to the class $\text{UL} \cap \text{co-UL}$. That is, the following conditions are equivalent:

1. f is computed by a logspace machine with an oracle from $\text{UL} \cap \text{co-UL}$.
2. f is computed by a logspace-uniform NC^1 circuit family with oracle gates for a language in $\text{UL} \cap \text{co-UL}$.
3. $f(x)$ has length bounded by a polynomial in $|x|$, and the set $\{(x, i, b) : \text{the } i^{\text{th}} \text{ bit of } f(x) \text{ is } b\}$ is in $\text{UL} \cap \text{co-UL}$.

Thus, if any of those conditions hold, we will say that “ f is computable in $\text{UL} \cap \text{co-UL}$ ”.

The important fact that the composition of two logspace-computable functions is also logspace-computable (see, e.g., [6]) carries over with an identical proof to the functions computable in L^C for any oracle C . Thus the class of functions computable in $\text{UL} \cap \text{co-UL}$ is also closed under composition. We make implicit use of this fact frequently when presenting our algorithms. For example, we may say that a colored labeling of a graph G is computable in $\text{UL} \cap \text{co-UL}$, and that, given such a colored labeling, a decomposition of the graph into layers is also computable in logspace, and furthermore, that – given such a decomposition of G into layers – an additional coloring of the smaller graphs is computable in $\text{UL} \cap \text{co-UL}$, etc. The reader need not worry that a logspace-bounded machine does not have adequate space to store these intermediate representations; the fact that the final result is also computable in $\text{UL} \cap \text{co-UL}$ follows from closure under composition. In effect, the bits of these intermediate representations are re-computed each time we need to refer to them.

Finally, we will consider AC^k circuits augmented with oracle gates for an oracle in $\text{UL} \cap \text{co-UL}$, which we denote by $\text{AC}^k(\text{UL} \cap \text{co-UL})$.

3 DFS in DAGs logspace reduces to Reachability

In this section, we observe that constructing the lexicographically-least DFS tree in a DAG G can be done in logspace given an oracle for reachability in G . But first, let us define what we mean by the lexicographically-first DFS tree in G :

► **Definition 1.** *Let G be a DAG, with the neighbours of the vertices given in some order in the input. (For example, with adjacency lists, we can consider the ordering in which the neighbors are presented in the list). Then the lexicographic first DFS traversal of G is the traversal with the (very natural) condition that the children of every vertex are explored in the order given in the input. For details, see the full version [3].*

The lemma we need is the following:

► **Lemma 2.** *Construction of lexicographic least DFSs tree in DAGs logspace reduces to reachability. In particular, DFS in general DAGs, planar DAGs, and planar DAGs with single source (SMPDs) lie in classes NL, $UL \cap \text{co-UL}$, and L respectively.*

The correctness of this lemma is shown by the proof of Theorem 11 of [10] for general DAGs. The extension for the other two classes is a consequence of planar reachability in $UL \cap \text{co-UL}$ [24] and of SMPD reachability in L [2]. We defer the details to the full version [3].

4 Overview of the Algorithm

The main algorithmic insight that led us to the current algorithm was a generalization of the layering algorithm that Hagerup developed for *undirected* graphs [14]. We show that this approach can be modified to yield a useful decomposition of *directed* graphs, where the layers of the graph have a restricted structure that can be exploited. More specifically, the strongly-connected components of each layer are what we call *meshes*, which enable us easily to construct paths (which will end up being paths in the DFS trees we construct) whose removal partitions the graph into significantly smaller strongly connected components.

The high-level structure of the algorithm is thus:

1. Construct a planar embedding of G .
2. Partition the planar graph G into layers (each of which is surrounded by a directed cycle).
3. Identify one such cycle C that has properties that will allow us to partition the graph into smaller weakly connected components.
4. Depending on which properties C satisfies, create a path p from the exterior face either to a vertex on C or to one of the meshes that reside in the layer just inside C . Removal of p partitions G into weakly connected components, where each strongly-connected component therein is smaller than G by a constant factor.
5. Let the vertices on this path p be v_1, v_2, \dots, v_k . The DFS tree will start with the path p , and append DFS trees for subgraphs G_1, G_2, \dots, G_k to this path, where G_i consists of all of the vertices that are reachable from v_i that are not reachable from v_j for any $j > i$. (This is obviously a tree, and it will follow that it is a DFS tree.) Further, decompose each G_i into a DAG of strongly-connected components. Build a DFS of that DAG, and then work on building DFS trees of the remaining (smaller) strongly-connected components.
6. Each of the steps above can be accomplished in $UL \cap \text{co-UL}$, which means that there is an AC^0 circuit with oracle gates from $UL \cap \text{co-UL}$ that takes G as input and produces the list of much smaller graphs G_1, \dots, G_k , as well as the path p that forms the spine of the DFS tree. We now recursively apply this procedure (in parallel) to each of these smaller graphs. The construction is complete after $O(\log n)$ phases, yielding the desired $AC^1(UL \cap \text{co-UL})$ circuit family.

In the exposition below, we first layer the graph in terms of clockwise cycles (which we will henceforth call red cycles), and obtain a decomposition of the original graph into smaller pieces. We then apply a nested layering in terms of counterclockwise cycles (which we will henceforth call blue cycles); ultimately we decompose the graph into units that are structured as a DAG, which we can then process using the tools from the earlier sections of the paper. The more detailed presentation follows.

4.1 Degree Reduction and Expansion

► **Definition 3** (of $\mathbf{Exp}^\circ(G)$ and $\mathbf{Exp}^\circ(G)$). *Let G be a planar digraph. The “expanded” digraph $\mathbf{Exp}^\circ(G)$ (respectively, $\mathbf{Exp}^\circ(G)$) is formed by replacing each vertex v of total degree $d(v) > 3$ by a clockwise (respectively, counterclockwise) cycle C_v on $d(v)$ vertices such that the endpoint of the i -th edge incident on v is now incident on the i -th vertex of the cycle.*

$\mathbf{Exp}^\circ(G)$ and $\mathbf{Exp}^\circ(G)$ each have maximum degree bounded by 3; i.e., they are *subcubic*. Next we define the clockwise (and counterclockwise) dual for such a graph and also a notion of distance.

Recall that for an undirected plane graph H , the dual (multigraph) H^* is formed by placing, for every edge $e \in E(H)$, a dual edge e^* between the face(s) on either side of e (see Section 4.6 from [12] for more details). Faces f of H and the vertices f^* of H^* correspond to each other as do vertices v of H and faces v^* of H^* .

► **Definition 4** (of Duals G° and G°). *Let G be a plane digraph, then the clockwise dual G° (respectively, counterclockwise dual G°) is a weighted bidirected version of the undirected dual of the underlying undirected graph of G in a way so that the orientation formed by rotating the corresponding directed edge of G in a clockwise (respectively, counterclockwise) way gets a weight of 1 and the other orientation gets weight 0. We inherit the definition of dual vertices and faces from the underlying undirected dual.*

► **Definition 5.** *For a plane subcubic digraph G , let f_0 be the external face. Define the type $\mathbf{type}^\circ(f)$ (respectively, $\mathbf{type}^\circ(f)$) of a face to be the singleton set consisting of the distance at which f lies from f_0 in G° : $\{d^\circ(f_0, f)\}$ (respectively, $\{d^\circ(f_0, f)\}$). Generalise this to edges e by defining $\mathbf{type}^\circ(e)$ (respectively $\mathbf{type}^\circ(e)$) as the set consisting of the union of the \mathbf{type}° (respectively, \mathbf{type}°) of the two faces adjacent to e , and to vertices v by defining as the $\mathbf{type}^\circ(v)$ (respectively $\mathbf{type}^\circ(v)$) union of the \mathbf{type}° (respectively, \mathbf{type}°) of the faces incident on the vertex v .*

The following is a direct consequence of subcubicity and the triangle inequality:

► **Lemma 6.** *In every subcubic graph G , the cardinality $|\mathbf{type}^\circ(x)|, |\mathbf{type}^\circ(x)|$ where x is a face, edge or a vertex is at least one and at most 2 and in the latter case consists of consecutive non-negative integers.*

Further, if $v \in V(G)$ is such that $|\mathbf{type}^\circ(v)| = 2$, then there exist unique $u, w \in V(G)$, such that $(u, v), (v, w) \in E(G)$ and $|\mathbf{type}^\circ(u, v)| = |\mathbf{type}^\circ(v, w)| = 2$.

For proof, see Appendix A.1.

► **Definition 7.** *For a plane subcubic graph G as above, we refer to vertices and edges with a type of cardinality two in G° (respectively, in G°) as red (respectively, blue) while the ones with a cardinality of one as white. The resulting colored graphs are called $\mathbf{red}(G)$ and $\mathbf{blue}(G)$ respectively.*

7:6 Depth-First Search in Directed Planar Graphs, Revisited

We will see later how to apply both the duals in G to get red and blue layerings of a given input graph.

Also note that a red (respectively blue) edge must have red (respectively blue) end points, as they are adjacent to the same faces as the edge between is.

We enumerate some properties of $\mathbf{red}(G)$, $\mathbf{blue}(G)$ (G is subcubic):

► **Lemma 8.**

1. Red vertices and edges in $\mathbf{red}(G)$ form disjoint clockwise cycles.
2. No clockwise cycle in $\mathbf{red}(G)$ consists of only white edges (and hence white vertices). Similar properties hold for $\mathbf{blue}(G)$.

Proof.

1. Firstly, note that a red edge must have red end point vertices, as they are adjacent to the same faces that the edge between them is adjacent to. It is immediate from Lemma 6 that if v is a red vertex, it has exactly one red incoming edge and one red outgoing edge, proving that they form disjoint cycles. Now consider a red cycle C . The type of each edge of C must be the same, since if there are two consecutive edges in C of different types, it would make the common vertex adjacent to at least three vertices of different types contradicting lemma 6. This means that the distance in G° of each face bordering the “outside” of C from the external face is one less than the distance of each face bordering the “inside” of C . But in any *counterclockwise* cycle, the distance in G° from the external face to both sides of C are the same (by the way distances are defined in G°). Thus C is clockwise.
2. Suppose C is a clockwise cycle. Consider the shortest path in G° from the external face to a face enclosed by C . From the Jordan curve theorem (Theorem 4.1.1 [12]), it must cross the cycle C . The edge dual to the crossing must be red. ◀

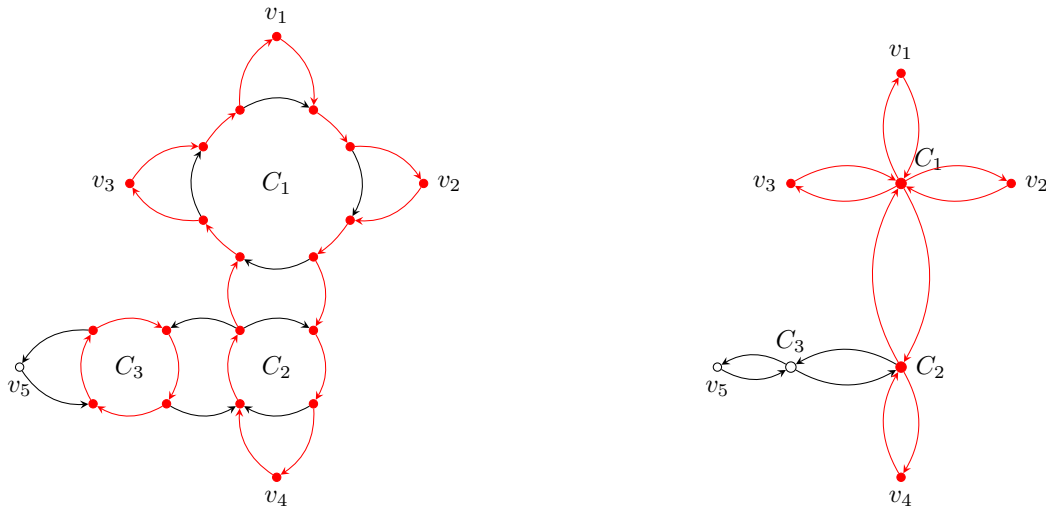
The definitions above, which apply only to subcubic plane graphs, can now be extended to a general plane graph G , by considering the subcubic graphs $\mathbf{Exp}^\circ(G)$ (and $\mathbf{Exp}^\circ(G)$). But first, we must make a simple observation about $\mathbf{red}(\mathbf{Exp}^\circ(G))$ (respectively about $\mathbf{blue}(\mathbf{Exp}^\circ(G))$).

► **Lemma 9.** *Let $v \in V(G)$ be a vertex of degree more than 3. Let C_v be the corresponding expanded cycle in $\mathbf{Exp}^\circ(G)$. Suppose at least one edge of C_v is white in $\mathbf{red}(\mathbf{Exp}^\circ(G))$ then there is a unique red cycle C that shares edges with C_v .*

Proof. First we note that C_v does not contain anything inside it since it is an expanded cycle. By lemma 8 we know that C_v has at least one red edge. Suppose it shares one or more edges with a red cycle R_1 . Since both cycles are clockwise and C_v has nothing inside, the cycle R_1 must enclose C_v . Now suppose there is another red cycle R_2 that shares one or more edges with C_v . Then R_2 must also enclose C_v . But two cycles cannot enclose a cycle whilst sharing edges with it without touching each other, which contradicts the above lemma that all red cycles in a subcubic graph are vertex disjoint. ◀

The last two lemmas allow us to consistently contract the red cycles in $\mathbf{red}(\mathbf{Exp}^\circ(G))$:

► **Definition 10.** *The colored graph $\mathbf{Col}^\circ(G)$ (respectively, $\mathbf{Col}^\circ(G)$) is obtained by labeling a degree more than 3 vertex $v \in V(G)$ as red iff the cycle C_v in $\mathbf{red}(\mathbf{Exp}^\circ(G))$ has at least one red edge and at least one white edge. Else the color of v is white. All the low degree vertices and edges of G inherit their colors from $\mathbf{red}(\mathbf{Exp}^\circ(G))$. The coloring of $\mathbf{Col}^\circ(G)$ is similar.*



■ **Figure 1** An example of contracting expanded cycles. The figure on right shows the graph after contracting the expanded cycles C_1, C_2, C_3 according to definition 10.

We can now characterize the colorings in the graph $\mathbf{Col}^\circ(G)$:

► **Lemma 11.** *The following hold:*

1. A red cycle in $\mathbf{Col}^\circ(G)$ is vertex disjoint from every red cycle contained in its interior.
2. Every 2-connected component of the red subgraph of $\mathbf{Col}^\circ(G)$ is a simple clockwise cycle.

We defer the proof to Section A.1.

Although the above lemmas have been proved for the clockwise dual, they also hold for counterclockwise dual with red replaced by blue.

4.2 Layering the colored graphs

► **Definition 12.** *Let $x \in V(\mathbf{Col}^\circ(G)) \cup E(\mathbf{Col}^\circ(G))$. Let $\ell^\circ(x)$ be one more than the minimum integer that occurs in $\mathbf{type}^\circ(x')$, for each $x' \in V(\mathbf{Exp}^\circ(G)) \cup E(\mathbf{Exp}^\circ(G))$ that is contracted to x . Further let $\mathcal{L}^k(\mathbf{Col}^\circ(G)) = \{x \in V(\mathbf{Col}^\circ(G)) \cup E(\mathbf{Col}^\circ(G)) : \ell^\circ(x) = k\}$. Similarly define, $\ell^\circ(x), \mathcal{L}^k(\mathbf{Col}^\circ(G))$. We call $\mathcal{L}^k(\mathbf{Col}^\circ(G))$ the k^{th} layer of the graph.*

See Fig 11 for an example. It is easy to see the following from Lemma 11:

► **Proposition 13.** *For every $x \in V(\mathbf{Col}^\circ(G)) \cup E(\mathbf{Col}^\circ(G))$ the quantity $\ell^\circ(x)$ is one more than the number of red cycles that strictly enclose x in $\mathbf{Col}^\circ(G)$. All the vertices and edges of a red cycle of $\mathbf{Col}^\circ(G)$ lie in the same layer $\mathcal{L}^{k+1}(\mathbf{Col}^\circ(G))$ for the enclosure depth k of the cycle.*

We had already noted above that the red subgraph of G had simple clockwise cycles as its biconnected components. We note a few more lemmas about the structure of a layer of G :

► **Lemma 14.** *We have:*

1. A red cycle in a layer $\mathcal{L}^{k+1}(\mathbf{Col}^\circ(G))$ does not contain any vertex/edge of the same layer inside it.
2. Any clockwise cycle in a layer consists of only red vertices and edges.

Dually, a blue cycle in a layer does not contain any vertex or edge of the same layer inside it.

► **Remark 15.** Notice that the conclusion in the second part of the lemma fails to hold if we allow cycles spanning more than one layer.

Proof. The first part is a direct consequence of proposition 13. For the second part we mimic the proof of the second part of Lemma 8. Consider a clockwise cycle $C \subseteq \mathcal{L}^{k+1}(\mathbf{Col}^\circ(G))$ that contains a white edge e . Every face adjacent to C from the outside must have $\mathbf{type}^\circ = k$ because C is contained in layer $k + 1$. Then the \mathbf{type}° of the faces on either side of e is the same and therefore must be k . Let f be a face enclosed by C that has $\mathbf{type}^\circ(f) = k$. Thus it must be adjacent to a face of $\mathbf{type}^\circ = k - 1$. But this contradicts that every face inside and adjacent to C must have \mathbf{type}° at least k . ◀

The lemmas above show that the strongly connected components of the red subgraph of a layer consist of red cycles touching each other without nesting, in a tree like structure. This prompts the following definition:

► **Definition 16.** For a red cycle $R \subseteq \mathcal{L}^k(\mathbf{Col}^\circ(G))$ we denote by G_R , the graph induced by vertices of $\mathcal{L}^{k+1}(\mathbf{Col}^\circ(G))$ enclosed by R .

Now we combine Definitions 10 and 12:

► **Definition 17.** Each vertex or edge $x \in V(G) \cup E(G)$ gets a red layer number $k + 1$ if it belongs to $\mathcal{L}^{k+1}(\mathbf{Col}^\circ(G))$ and a blue layer number $l + 1$, if it belongs to $\mathcal{L}^{l+1}(\mathbf{Col}^\circ(G_R))$ where $R \subseteq \mathcal{L}^k(\mathbf{Col}^\circ(G))$ is the red cycle immediately enclosing x .

Moreover this defines the colored graph $\mathbf{Col}(G)$ by giving x the color red if it is red in $\mathbf{Col}^\circ(G)$ and/or blue in $\mathbf{Col}^\circ(G_R)$ (notice it could be both red and blue) and lastly white if it is white in both the graphs. In this case, we say that x belongs to sublayer $\mathcal{L}^{k+1,l+1}(\mathbf{Col}(G))$.

By Proposition 13, we can also say that a sublayer $\mathcal{L}^{k+1,l+1}(\mathbf{Col}(G))$ thus consists of edges/vertices that are strictly enclosed inside k red cycles and inside l blue cycles that are contained *inside* the first enclosing red cycle.

We'll see some observations and lemmas regarding the structure of a sublayer now.

Since every edge/vertex in $\mathcal{L}^{k+1,l+1}(\mathbf{Col}(G))$ has the same red AND blue layer number, it is clear that there can be no nesting of colored cycles. Also we have:

► **Lemma 18.** Every clockwise cycle in a sublayer $\mathcal{L}^{k+1,l+1}(\mathbf{Col}(G))$ consists of all red edges and vertices and any every counterclockwise cycle in the sublayer consists of all blue vertices and edges. (Some edges/vertices of the cycle can be both red as well as blue)

Proof. This is a direct consequence of Lemma 14 applied to the sublayer $\mathcal{L}^{k+1,l+1}(\mathbf{Col}(G))$, which is a (counterclockwise) layer in graph G_R for some red cycle R . ◀

Thus we can refer to clockwise cycles and counterclockwise cycles as red and blue cycles respectively.

► **Definition 19.** For a red or blue colored cycle C of layer $\mathcal{L}^{k,l}(\mathbf{Col}(G))$, we denote by G_C the graph induced by vertices of $\mathcal{L}^{k',l'}(\mathbf{Col}(G))$ enclosed by C , where $\{k',l'\}$ is $\{k + 1, 1\}$ or $\{k, l + 1\}$ according to whether C is a red or a blue cycle respectively.

Note that:

► **Proposition 20.** Two cycles of the same color in $\mathcal{L}^{k+1,l+1}(G)$ cannot share an edge.

This is since neither is enclosed by the other as they belong to the same layer, and as they also have the same orientation. Cycles of different colors can share edges but we note:

► **Lemma 21.** *Two cycles of a sublayer $\mathcal{L}^{k+1,l+1}(\text{Col}(G))$ can only share one contiguous segment of edges.*

Proof. Let a red cycle R and a blue cycle B in a sublayer share two vertices u, v but let the paths $R(u, v), B(u, v)$ in the two cycles be disjoint. Notice that the graph $(R \setminus R(u, v)) \cup B(u, v)$ is also a clockwise cycle that encloses the edges of $R(u, v)$ contradicting the first part of Lemma 14. ◀

We consider the strongly connected components of a sublayer and note the following lemmas regarding them:

► **Lemma 22.** *The trivial strongly connected components of a sublayer (those that consist of a single vertex) are white vertices. The non-trivial strongly connected components of a sublayer have the following properties:*

1. *Every vertex/edge in them is blue or red (possibly both).*
2. *Every face, except possibly the outer face, is a directed cycle.*
3. *Every face other than the outer face has at least one edge adjacent to the outer face.*

We defer the proof to Section A.1. The strongly connected components of a sublayer hence consist of intersecting red and blue facial cycles, with every face having at least one boundary edge adjacent to the outer face of the component.

► **Definition 23.** *We call the strongly connected components of a sublayer $\mathcal{L}(k, l)$ meshes.*

5 Mesh Properties

► **Definition 24.** *Given a subgraph H of G embedded in the plane, we define the closure of H , denoted by \tilde{H} , to be the induced graph on the vertices of H together with the vertices of G that lie in the interior of faces of H (except for the outer face of H).*

For convenience, we call a face of a graph that is not the outer face an *internal face*.

From Lemmas 18 and 22, we have a bijection: every face of a mesh, except possibly its outer face, is a directed cycle, and every directed cycle in a mesh is the boundary of a face of the mesh.

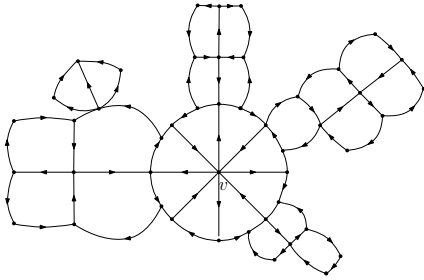
► **Definition 25.** *Let $0 < \alpha < 1$. An α separator of a digraph H that is a subgraph of a digraph G is a set of vertices of H whose removal from H separates \tilde{H} into subgraphs, where no strongly connected component has size greater than $\alpha|G|$. A path separator is a sequence of vertices $\langle v_1, \dots, v_n \rangle$ that is a separator and also is a directed path.*

► **Definition 26.** *Let G be a graph and let M be a mesh in a sublayer G . For an internal face f of M , we define $wt(f)$ to be $|V(\tilde{f})|$. Let $wt(H)$ where H is a subgraph of M be defined as $|V(\tilde{H})|$.*

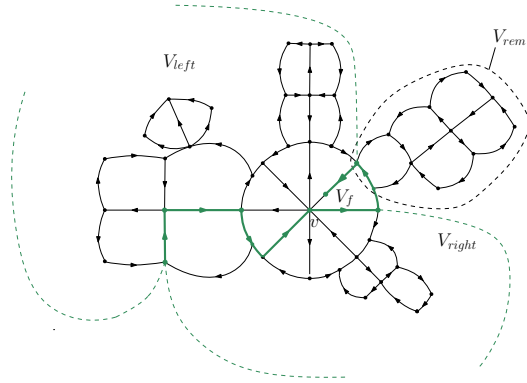
► **Definition 27.** *For a mesh M , we call a vertex that is adjacent to the outer face of M an external vertex, and a vertex that is not adjacent to the outer face an internal vertex. Also, we call vertices of degree more than two junction vertices.*

*If $p = \langle v_1, v_2, \dots, v_k \rangle$ is a directed path such that v_2, \dots, v_{k-1} are all vertices of degree two, but v_1, v_k have degree more than two, then we call p a **segment**. We call v_k the out junction neighbour of v_1 and v_1 the in junction neighbour of v_k .*

*We call a **segment** with all edges adjacent to the outer face an external **segment**, and a **segment** with no edge adjacent to the outer face an internal **segment**. If the end points of an internal **segment** are both internal vertices also, we call the segment an **i-i-segment**.*



■ **Figure 2** An example of a mesh.



■ **Figure 3** An example of a path separator. The vertex v is a central node, and the green path is a separator.

The rest of this section is devoted to a proof of the following, which asserts that we can construct a path separator in a mesh, assuming that no internal face of the mesh is too large.

► **Lemma 28.** *Suppose $wt(f) < wt(G)/12$ holds for every internal face f of a mesh M that is a subgraph of G . Then from any external vertex r of M , we can find (in $UL \cap co-UL$) an $\frac{11}{12}$ path separator of M , starting at r .*

The high level idea is that using a clique sum decomposition of 2, 3-cliques (see figure 9) we find a “central” vertex v in the mesh M , such that we can find a path from the external vertex r to v , and then extend the path around one of the faces adjacent to v to get the path separator (all faces are directed cycles by lemma 22). Because every face touches the outer face and weight of every face is small by the hypothesis of the lemma, we can always find a face adjacent to v to encircle such that removing the path leaves no large (weakly) connected component. The vertices of M with degree two (in-degree 1 and out-degree 1 because M is strongly connected) are not important since they can be seen as just “subdivision” vertices. Now we will look at the structure of a mesh around an internal junction vertex, and the way the rest of the mesh is attached to that structure. Also, we state here that we will abuse the notion of 3-connected components by ignoring the non-junction vertices for convenience.

► **Lemma 29.** *If v is an internal junction vertex of a mesh and e_1, \dots, e_k are the edges adjacent to v in the cyclic order of embedding, then the edges alternate in directions i.e. if e_1 is outgoing from v , then e_2 is incoming to v and e_3 is outgoing and so on. Consequently, v has even degree (at least 4).*

► **Definition 30.** *Let v be an internal junction vertex of degree $2d$ in a mesh M , and let its junction neighbours be $(u_1, w_1, u_2, w_2, \dots, u_d, w_d)$ in clockwise order starting from edge $\langle u_1, v \rangle$ (the w_i 's are out neighbours, and u_i 's the in neighbours, since junction neighbours alternate).*

*Every adjacent pair of edges incident to v borders a face that is not the outer face. Let $f_{u,v,w}$ denote the face bordered by v and the junction neighbours u and w of v which are adjacent in cyclic order around v . The boundary of $f_{u,v,w}$ can be written as three disjoint parts (except for endpoints), **segment** (u, v) + **segment** (v, w) + **petal** $_{w,u}$, where the third part denotes a simple path from w to u along the face boundary. We will use the notation $petal_{w,u}$ to denote the corresponding boundary for any face $f_{u,v,w}$ adjacent to v . We define **flower** (v) as $\bigcup \{ \text{vertices on boundary of faces adjacent to } v \}$ (See figure 4).*

We note the following property of petals whose proof is deferred to AppendixA.2.

► **Proposition 31.** *For all adjacent junction neighbour pairs w_i, u_j of internal vertex v , petal_{w_i, u_j} are disjoint, except possibly the end points.*

For an internal junction vertex v , the union of the petals around **flower**(v) thus form an undirected cycle around v , with at least four alternations in directions. Now we define bridges of the cycle, which roughly, are components of M we get after removing **flower**(v), leaving the points of attachment intact. We use the formal definition of bridges from [25]:

► **Definition 32.** *For a subgraph H of M , a vertex of attachment of H is a vertex of H that is incident with some edge of M not belonging to H . Let J be an undirected cycle of M . We define a **bridge** of J in M as a subgraph B of M with the following properties:*

1. *each vertex of attachment of B is a vertex of J .*
2. *B is not a subgraph of J .*
3. *no proper subgraph of B has both the above properties.*

*We denote by **2-bridge**, bridges with exactly two vertices of attachment to the specified cycle, and by **3-bridge**, bridges with three or more vertices of attachment.*

Note that for the cycle formed by petals of **flower**(v), the vertex v along with paths leading to/ coming from **flower**(v) also form a bridge, but we call that a trivial bridge and do not take it into consideration.

► **Lemma 33.**

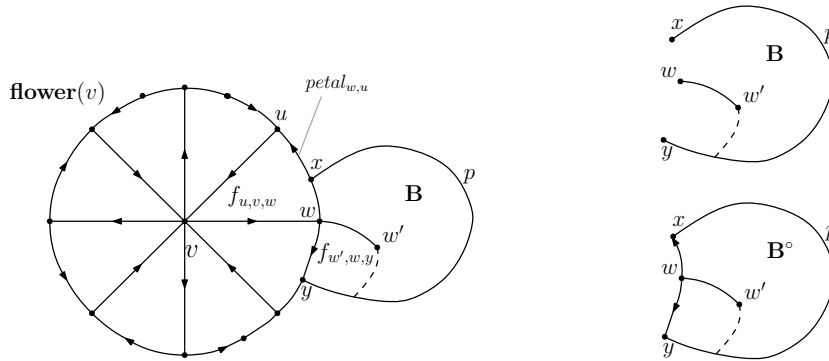
1. *The vertices of attachment of a **2-bridge** of **flower**(v) must both lie on one **petal** of **flower**(v).*
2. *The vertices of attachment of a **3-bridge** of **flower**(P) can lie on one or, at most two adjacent petals. Moreover, in the latter case the junction neighbour of v common to both petals must be a vertex of attachment of the **3-bridge**.*
3. *For an internal vertex v , and an external vertex r of M , let $p = \langle r, \dots, u_1, v \rangle$ be a simple path from r to v , where u_1 is an in junction neighbour of v . Let the other junction neighbours of v be named as in Definition 30 in cyclic order from u_1 . For $j \in \{i, i+1\}$, consider an extended path of p , $p_{w_i, u_j} = \langle r, \dots, u_1, v, w_i \rangle + \text{petal}_{w_i, u_j} + \langle u_j, \dots, v \rangle$, excluding the last edge incident to v in the sequence. That is, p_{w_i, u_j} goes from r to v , then to an out junction neighbour w_i , and then wraps around f_{u_j, v, w_i} by taking petal_{w_i, u_j} and then the segment back towards v from u_j . If there is a bridge of **flower**(v) of which u_1 is a point of attachment and also includes the edge of p incoming to u_1 , we denote it by B_{in} . The set $V(\tilde{M}) \setminus V(p_{w_i, u_j})$ can be partitioned into four disconnected parts, say V_{left} and V_{right} , V_f , V_{rem} such that:*

$$\begin{aligned} V_{left} = & (\{\tilde{f}_{u_1, v, w_1} \cup \tilde{f}_{u_2, v, w_1} \cup \tilde{f}_{u_2, v, w_2} \dots \cup \tilde{f}_{u_i, v, w_{i-1}}\} \cup \{\tilde{f}_{u_i, v, w_i} \text{ if } j = i + 1\}) \\ & \cup \{\text{vertices in closure of bridges attached to the petals of these faces, excluding } B_{in}\} \\ & \cup \{\text{the "left" part of } B_{in}.(\text{See figure 8})\} \setminus V(p_{w_i, u_j}) \end{aligned}$$

$$\begin{aligned} V_{right} = & (\{\tilde{f}_{u_i, v, w_{i+1}} \cup \tilde{f}_{u_{i+2}, v, w_{i+1}} \dots \cup \tilde{f}_{u_d, v, w_d}\} \cup \{\tilde{f}_{u_{i+1}, v, w_i} \text{ if } j = i\}) \\ & \cup \{\text{vertices in closure of bridges attached to petals petals these faces, excluding } B_{in}\} \\ & \cup \{\text{the "right" part of } B_{in}.(\text{See figure 8})\} \setminus V(p_{w_i, u_j}) \end{aligned}$$

$$V_f = \tilde{f}_{u_j, v, w_i} \setminus V(p_{w_i, u_j})$$

$$\begin{aligned} V_{rem} = & \left(\bigcup \{\text{vertices in closure of all bridges that have vertices} \right. \\ & \left. \text{of attachment only in } \text{petal}_{w_i, u_j}\} \right) \setminus V(p_{w_i, u_j}). \end{aligned}$$



■ **Figure 4** A vertex v and $\text{flower}(v)$. B is a **bridge** with two points of attachment x, y on two different petals of $\text{flower}(v)$. On the right are drawn the **bridge** B itself, and its closed version B° . The only way the boundary of $f_{w',w,y}$ can have an external edge is if it touches B , making w a point of attachment of B also.

such that there is no undirected path between any vertex of one of these four sets to any vertex of another. The path p_{w_i, u_i} is therefore a path separator that gives these components.

We introduce another notation for an extension of a bridge:

► **Definition 34.** For a bridge B of $\text{flower}(v)$, we define B° as B along with segments of $\text{flower}(v)$ that lie between consecutive vertices of attachment of B . We call this the **closed bridge** of B .

Now we will give definitions/lemmas regarding the “internal structure” of meshes, that will be useful to define the “center” of a mesh.

► **Definition 35.** For a mesh M , we call its **internal-skeleton**, denoted by $I(M)$, the induced subgraph on the vertices of **i-i-segments** of M . (See figure 6)

► **Lemma 36.**

1. For a mesh M , the graph $I(M)$ is a forest.
2. If H is a 3-connected induced subgraph of M (ignoring subdivision vertices), then $I(H)$ is a tree.

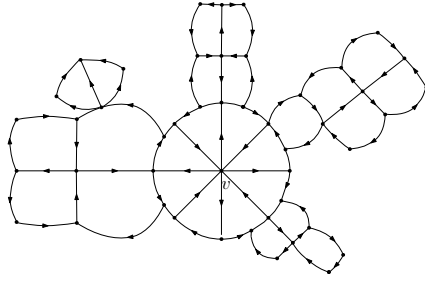
We state a well-known proposition about a vertex separator in a tree T with weighted nodes, without the proof.

► **Proposition 37.** Suppose T is a tree with each node having a weight assigned to it. Let $wt(T')$ denote sum of weights of each node in a subgraph T' of T . Then there exists a node v_c or a pair of adjacent nodes v_{c_1}, v_{c_2} , such that after removing it (or them in case of a pair), no connected component in the remaining forest has weight more than $\frac{1}{2}wt(T)$.

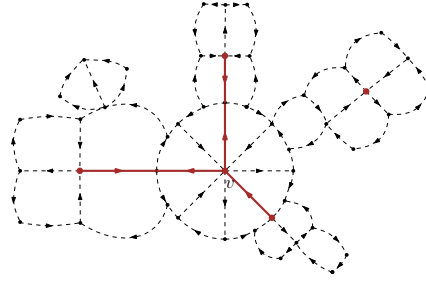
We will next give a procedure to define a “center” of a mesh.

► **Definition 38.** For a mesh M , let T_M denote the tree obtained by the 1,2-clique sum decomposition of M . The nodes of T_M are of two types, clique nodes (cut vertices or separating pairs), and piece nodes, which are either 3-connected parts or cycles. Every piece node is adjacent to a clique node and vice-versa. (See [9, Section 3.1] for background about this decomposition.)

Consider the $\frac{1}{2}$ separator node of T_M as described in Proposition 37. If it is a separating pair, a cut vertex, or a face cycle, we call that subgraph the **center** of M .



■ **Figure 5** An example of a mesh.



■ **Figure 6** The internal skeleton of the mesh. One of its components is a single node.

If it is a 3-connected node P , look at its internal skeleton $I(P)$. We construct a new graph $I'(P)$ which is isomorphic to $I(P)$ but has edges directed differently. Let u, v be two adjacent internal junction vertices of M . To give direction to a **segment** (u, v) in $I'(P)$, we consider the unique **bridge** B of $\text{flower}(u)$ that contains v as a point of attachment; we denote the **closed bridge** of B by $\mathbf{B}_u^\circ(v)$. $\mathbf{B}_v^\circ(u)$ is defined analogously. We orient (u, v) in the direction of the heavier of $\mathbf{B}_u^\circ(v)$ and $\mathbf{B}_v^\circ(u)$ (breaking ties arbitrarily), where the weights of $\mathbf{B}_u^\circ(v), \mathbf{B}_v^\circ(u)$ are $|\widetilde{\mathbf{B}}_u^\circ(v)|$ and $|\widetilde{\mathbf{B}}_v^\circ(u)|$, respectively.

The **center** of M is defined to be $\text{flower}(v)$ in this case, where v is the sink node of $I'(P)$.

We show why $I'(P)$ cannot have more than one sink.

► **Lemma 39.** *The tree $I'(P)$ defined above will have exactly one sink vertex.*

► **Lemma 40.** *If the center of M is $\text{flower}(v)$, and w is an out neighbor of v , then $wt(\mathbf{B}_v^\circ(w)) \leq \frac{1}{2}(wt(\widetilde{M}) - wt(V_{rem}(u, w)))$, where u is either of the two in neighbors of v that are adjacent to w around $\text{flower}(v)$.*

Proof. Since the center is $\text{flower}(v)$, we have that $wt(\mathbf{B}_v^\circ(w)) \leq wt(\mathbf{B}_w^\circ(v))$. But $V_{rem}(u, w)$ has empty intersection with each of $\mathbf{B}_v^\circ(w)$ and $\mathbf{B}_w^\circ(v)$. Thus $wt(\mathbf{B}_v^\circ(w)) + wt(\mathbf{B}_w^\circ(v)) \leq wt(\widetilde{M}) - wt(V_{rem}(u, w))$. The lemma follows. ◀

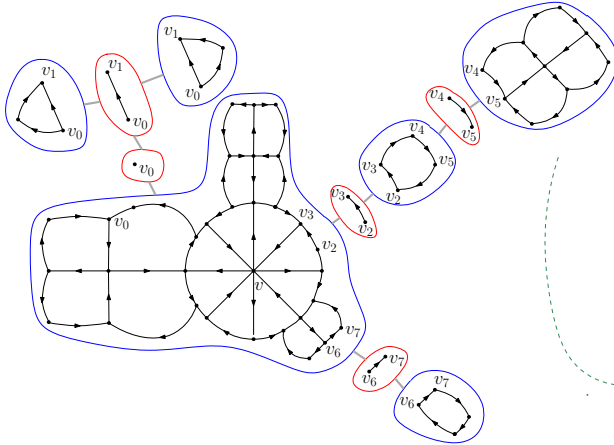
► **Lemma 41.**

1. *If the center of M is not of the form $\text{flower}(v)$ where v is an internal node of a 3-connected component, then removing it from \widetilde{M} disconnects \widetilde{M} into weakly connected components, each with weight less than $\frac{1}{2}wt(\widetilde{M})$.*
2. *If the center of M is $\text{flower}(v)$ for an internal node v of a 3-connected component P , then on removing $\text{flower}(v)$ from \widetilde{M} , no weakly connected component has weight more than $\frac{1}{2}wt(\widetilde{M})$.*

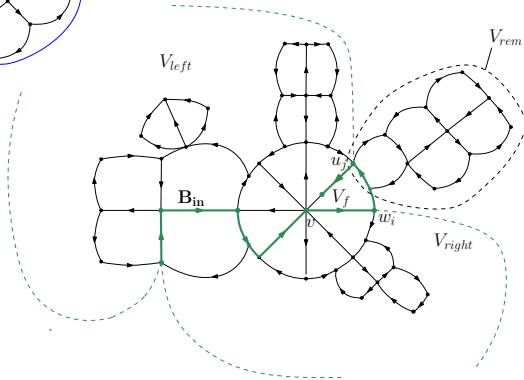
Proof.

1. This follows from the vertex separator lemma for trees with weighted vertices.
2. This follows from the v being the sink node of $I'(P)$. ◀

► **Lemma 42.** *For every possible path p_{w_i, u_j} around v as defined in Lemma 33, V_{rem} consists of a disjoint union of weakly-connected components, each of which has weight $\leq \frac{1}{2}(wt(M))$. For a path p_{w_i, u_j} (where $j \in \{i, i+1\}$) we sometimes use the notation $V_{rem}(w_i, u_j)$ to specify the petal where the bridges of V_{rem} are attached.*



■ **Figure 7** The tree decomposition of the mesh using 1,2-clique sums. The nodes encircled red are clique separator nodes.



■ **Figure 8** An example of a path separator. The vertex v is a central node, and the green path is a separator.

5.1 Mesh Separator Algorithm

Now we give the algorithm to find an α separator in a mesh $M(G)$, assuming the hypothesis of Lemma 28.

1. Find the decomposition tree, T_M of M with 2-cliques and 1-cliques as the separating sets.
2. Find the **center** of the mesh M . It will either be a cut vertex, a separating pair, a cycle, or **flower**(v) for some internal vertex v .
3. If it is a cut vertex, we just find a path from the root r to it. If it is a separating pair (u, v) , both the vertices must lie on a same face, which is a directed cycle. In both this case, and also the case in which the **center** is a cycle, find a path from the root to any vertex of the face that touches it the first time, and then extend the path by encircling the cycle.
4. If it is **flower**(v) for some internal vertex v , find a path $p = \langle r, \dots, u_1, v \rangle$ to v . Let the junction neighbours of v in clockwise order starting from (u_1, v) , be $w_1, u_2, w_2, \dots, w_d$, with the w 's being out junction neighbours and the u 's being in junction neighbours. Starting clockwise from segment $\langle u, v \rangle$, find the first index i and $j \in \{i, i + 1\}$ s.t. after removing the extended path p_{w_i, u_j} , (defined in Lemma 33) the remaining strongly connected components are smaller than $\frac{11}{12}wt(G)$.

The algorithm above can clearly be implemented in logspace with an oracle for planar reachability, and thus it can be implemented in $UL \cap co-UL$.

It remains to show that the “first i ” mentioned in the final step actually exists. For the proof see Appendix A.3:

► **Lemma 43.** *If the center of M is **flower**(v) for some internal vertex v , then there will always exist an adjacent face f_{u_i, v, w_i} s.t. the path p_{w_i, u_i} is a $\frac{11}{12}$ -separator.*

6 Path separator in a planar digraph

Having seen how to construct a path separator in a **mesh**, we now show how to use that to construct an $\frac{11}{12}$ separator in any planar digraph.

1. Given a graph G , first embed the graph so that the root r lies on the outer face. Through the root, draw a virtual directed cycle C_0 that encloses the entire graph, and orient it, say clockwise. Find the layering described in Section 4 and output it on a transducer. Cycle C_0 will be colored red and will be in the sublayer $(0, 0)$.

2. In the laminar family of red/blue cycles, find the cycle C s.t. $wt(C)$ is more than $|G|/12$, but no colored cycle C' in the interior of C has the same property. Such a cycle will clearly exist (it could be the virtual cycle C_0). Let the sublayer of C be (k, l) .
3. Find a path p from the root r to any vertex r_C of the cycle C such that no other vertex of C is in the path. As seen above in Lemma 22, the graph in the interior of C and belonging to the immediately next sublayer $((k+1, l)$ if C is clockwise and $(k, l+1)$ if C is counter-clockwise) is a DAG of meshes. There are two cases possible:
 - a. The graph \tilde{C} has no strongly connected components of weight larger than $|G|/12$. In this case we simply extend the path p from r_C by encircling the cycle C till the last vertex and stop.
 - b. The graph \tilde{C} has a strongly connected component of weight more than $|G|/12$. In this case, we extend p from r_C by encircling C till the last vertex u on C that can reach any such component M_C . Then extend the path from u to any vertex of M_C and apply the mesh separator lemma (Lemma 28) to obtain the desired separator. (Observe that M_C satisfies the hypothesis of Lemma 28.)

► **Lemma 44.** *The path p obtained by the above procedure is an $\frac{11}{12}$ separator.*

Proof. We look at the two cases:

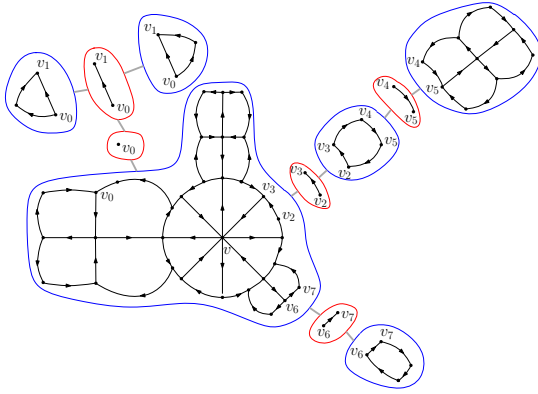
1. In this case it is clear that the interior and exterior of cycle C are disconnected by p . The exterior of C has size $\leq \frac{11}{12}|G|$ (by definition of C), and in its interior every strongly-connected component has weight at most $|G|/12$. Thus this satisfies the definition of an $\frac{11}{12}$ separator.
2. We took the last edge in C from r_C that can reach the mesh M_C , and extended the path to M_C . Thus after removing p , one weakly-connected component consists of the exterior of G , along with (possibly) some vertices in the interior of C that cannot reach any “large” mesh in the interior. Since M_C has weight greater than $\frac{1}{12}|G|$, no strongly-connected component embedded outside of M_C can have weight more than $\frac{11}{12}|G|$. Also, after removing path p , Lemma 28 guarantees that no other strongly-connected component will have weight more than $\frac{11}{12}|G|$. Thus this is an $\frac{11}{12}$ separator.

Hence overall we can guarantee an $\frac{11}{12}$ path separator in G . ◀

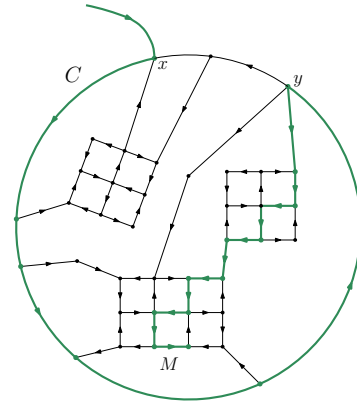
7 Building a DFS tree using path separators

We give a recursive divide and conquer algorithm for DFS:

1. Given a planar drawing of G and a root vertex on the outer face r , find an $\frac{11}{12}$ path separator $p = \langle r, v_1, v_2, \dots, v_k \rangle$. Path p is included in the DFS tree.
2. Let $R(v)$ denote the set of vertices of G reachable from v . Now for every vertex v_i in p compute in parallel: $R'(v_i) = R(v) \setminus (\bigcup_{j=i+1}^k R(v_j))$. Our DFS will correspond to first traveling along p to v_k , doing DFS on $R(v_k)$, and then while backtracking on p , do DFS on $R'(v_i)$ for i from $k-1$ down to 1. Given G , the encodings of p and $R'(v_i)$ can all be computed in $\text{AC}^0(\text{UL} \cap \text{co-UL})$.
3. For any v_i , $R'(v_i)$ can be written as a DAG of SCCs (strongly connected components), where each SCC is smaller than $\frac{11}{12}|G|$. In $\text{AC}^0(\text{UL} \cap \text{co-UL})$ we can compute this DAG and we can compute an encoding of the tuple (i, M, v) where M is a SCC in $R'(v_i)$ and v is a vertex in M . Recursively, in parallel, we compute a DFS tree of M for each tuple (i, M, v) , using v as the root. Now we need to show how to sew together (some of) these trees, to form a DFS tree for G with root r .



■ **Figure 9** The tree decomposition of the mesh using 1,2-clique sums. The nodes encircled red are clique separator nodes.



■ **Figure 10** The cycle C is a cycle satisfying the property stated in step 2 of the algorithm. The mesh M in the next sublayer is heavy, so we find a path from the last vertex on C that can reach M (in this case y), and then apply the algorithm of previous section on M .

4. Given a triple (i, M, v) , let x_0, x_1, \dots, x_r be the order in which the vertices of M appear in a DFS traversal where the root $x_0 = v$. Our DFS will correspond to first following the edges from x_0 that lead to other SCCs in $R(v_i)$. (No vertex reachable in this way can reach any x_j , or else that vertex would also be in M .) And then we will move on to x_1 and repeat the process, etc. Thus let $R''_{i,M,v}(x_j) = (R'(x_j) \setminus M) \setminus (\bigcup_{k < j} R'(x_k))$. Our DFS tree is composed by computing a DFS tree T of the DAG of meshes (considering each mesh to be a vertex) using the algorithm of Section 3. A logspace machine can do a DFS traversal of T , starting with the node containing v_i as the root, and using (as auxiliary information) the DFS tree that was computed for (i, M, v_i) . If this traversal contains an edge (M, M') (where M and M' are SCCs in $R'(v_i)$), then there is exactly one j such that there is an edge from x_j in the DFS tree for (i, M, v_i) to a vertex (call it $v_{M'}$) in $M' \cap R''_{i,M,v}(x_j)$. [Namely, x_j is the first vertex in this tree that has an edge to M' .] The edge from x_j to $v_{M'}$ will be in our DFS tree, as will the DFS tree that was computed for $(i, M', v_{M'})$. We then continue the traversal of T , and process each node of the DAG in the same way. All of this can be accomplished in $\text{AC}^0(\text{UL} \cap \text{co-UL})$.
5. The final DFS tree for R_i consists of all of the edges that appear in the trees for tuples (i, M, v) that were utilized in the traversal of T . The tree for G consists of p together with the trees for each R_i .

References

- 1 Alok Aggarwal, Richard J. Anderson, and Ming-Yang Kao. Parallel depth-first search in general directed graphs. *SIAM J. Comput.*, 19(2):397–409, 1990. doi:10.1137/0219025.
- 2 Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Planar and grid graph reachability problems. *Theory of Computing Systems*, 45(4):675–723, 2009. doi:10.1007/s00224-009-9172-z.
- 3 Eric Allender, Archit Chauhan, and Samir Datta. Depth-first search in directed graphs, revisited. Technical Report TR20-074, Electronic Colloquium on Computational Complexity (ECCC), 2020.

- 4 Eric Allender, Archit Chauhan, Samir Datta, and Anish Mukherjee. Planarity, exclusivity, and unambiguity. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:39, 2019.
- 5 Eric Allender, Klaus Reinhardt, and Shiyu Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59(2):164–181, 1999.
- 6 Sanjeev Arora and Boaz Barak. *Computational Complexity, a modern approach*. Cambridge University Press, 2009.
- 7 Tetsuo Asano, Taisuke Izumi, Masashi Kiyomi, Matsuo Konagaya, Hirotaka Ono, Yota Otachi, Pascal Schweitzer, Jun Tarui, and Ryuhei Uehara. Depth-first search using $O(n)$ bits. In Hee-Kap Ahn and Chan-Su Shin, editors, *Proc. 25th International Symposium on Algorithms and Computation (ISAAC)*, volume 8889 of *Lecture Notes in Computer Science*, pages 553–564. Springer, 2014. doi:10.1007/978-3-319-13075-0_44.
- 8 Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *TOCT*, 1(1):4:1–4:17, 2009. doi:10.1145/1490270.1490274.
- 9 Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar graph isomorphism is in log-space. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC)*, pages 203–214, 2009. doi:10.1109/CCC.2009.16.
- 10 Pilar de la Torre and Clyde P. Kruskal. Fast parallel algorithms for all-sources lexicographic search and path-algebra problems. *J. Algorithms*, 19(1):1–24, 1995. doi:10.1006/jagm.1995.1025.
- 11 Pilar de la Torre and Clyde P. Kruskal. Polynomially improved efficiency for fast parallel single-source lexicographic depth-first search, breadth-first search, and topological-first search. *Theory Comput. Syst.*, 34(4):275–298, 2001. doi:10.1007/s00224-001-1008-4.
- 12 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate texts in mathematics*. Springer, 2016.
- 13 Amr Elmasry, Torben Hagerup, and Frank Kammer. Space-efficient basic graph algorithms. In *Proc. 32nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 30 of *LIPICs*, pages 288–301. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.STACS.2015.288.
- 14 Torben Hagerup. Planar depth-first search in $O(\log n)$ parallel time. *SIAM J. Comput.*, 19(4):678–704, 1990. doi:10.1137/0219047.
- 15 Torben Hagerup. Space-efficient DFS and applications to connectivity problems: Simpler, leaner, faster. *Algorithmica*, 82(4):1033–1056, 2020. doi:10.1007/s00453-019-00629-x.
- 16 Taisuke Izumi and Yota Otachi. Sublinear-space lexicographic depth-first search for bounded treewidth graphs and planar graphs. In *Proc. 47th International Colloquium on Automata, Languages and Programming (ICALP)*, *LIPICs*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. to appear.
- 17 B. Jenner and B. Kirsig. Alternierung und Logarithmischer Platz. Dissertation, Universität Hamburg, 1989.
- 18 Ming-Yang Kao and Philip N. Klein. Towards overcoming the transitive-closure bottleneck: Efficient parallel algorithms for planar digraphs. *Journal of Computer and System Sciences*, 47(3):459–500, 1993. doi:10.1016/0022-0000(93)90042-U.
- 19 Maxim Naumov, Alysson Vrieling, and Michael Garland. Parallel depth-first search for directed acyclic graphs. In *Proc. 7th Workshop on Irregular Applications: Architectures and Algorithms*, pages 4:1–4:8, 2017. doi:10.1145/3149704.3149764.
- 20 John H. Reif. Depth-first search is inherently sequential. *Inf. Process. Lett.*, 20(5):229–234, 1985. doi:10.1016/0020-0190(85)90024-9.
- 21 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.
- 22 Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM J. Comput.*, 29(4):1118–1131, 2000. doi:10.1137/S0097539798339041.
- 23 Raghunath Tewari and N. V. Vinodchandran. Green’s theorem and isolation in planar graphs. *Inf. Comput.*, 215:1–7, 2012. doi:10.1016/j.ic.2012.03.002.

- 24 Thomas Thierauf and Fabian Wagner. The isomorphism problem for planar 3-connected graphs is in unambiguous logspace. *Theory Comput. Syst.*, 47(3):655–673, 2010. doi:10.1007/s00224-009-9188-4.
- 25 W. T. Tutte. Separation of vertices by a circuit. *Discrete Mathematics*, 12(2):173–184, 1975.
- 26 H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York Inc., 1999. doi:10.1007/978-3-662-03927-4.

A Appendix: Omitted details

A.1 Proofs from Section 4

We first need a simple lemma:

► **Lemma 45.** *Suppose (f_1, f_2) is a dual edge with weight 1 (and (f_2, f_1) is of weight 0) then, $d^\circ(f_0, f_1) \leq d^\circ(f_0, f_2) \leq d^\circ(f_0, f_1) + 1$.*

Proof. From the triangle inequality $d^\circ(f_0, f_1) \leq d^\circ(f_0, f_2) + d^\circ(f_2, f_1) = d^\circ(f_0, f_2)$. Similarly, $d^\circ(f_0, f_2) \leq d^\circ(f_0, f_1) + d^\circ(f_1, f_2) \leq d^\circ(f_0, f_1) + 1$. ◀

Proof of Lemma 6. Since each vertex $v \in V(G)$ of a subcubic graph is incident on at most 3 faces the only case in which $|\mathbf{type}^\circ(v)| > 2$ corresponds to three distinct faces f_1, f_2, f_3 being incident on a vertex. But here the undirected dual edges form a triangle such that in the directed dual the edges with weight 1 are oriented either as a cycle or acyclically. In the former case by three applications of the first half of Lemma 45 we get that $d^\circ(f_0, f_1) \leq d^\circ(f_0, f_2) \leq d^\circ(f_0, f_3) \leq d^\circ(f_0, f_1)$, hence all 3 distances are the same. Therefore $|\mathbf{type}^\circ(v)| = 1$.

In the latter case, suppose the edges of weight 1 are $(f_1, f_2), (f_2, f_3), (f_1, f_3)$, then by Lemma 45 we get: $d^\circ(f_0, f_1) \leq d^\circ(f_0, f_2), d^\circ(f_0, f_3) \leq d^\circ(f_0, f_1) + 1$. Thus, both $d^\circ(f_0, f_2), d^\circ(f_0, f_3)$ are sandwiched between two consecutive values $d^\circ(f_0, f_1), d^\circ(f_0, f_1) + 1$. Hence $d^\circ(f_0, f_1), d^\circ(f_0, f_2), d^\circ(f_0, f_3)$ must take at most two distinct values, and thus $|\mathbf{type}^\circ(v)| \leq 2$. Moreover either $\mathbf{type}^\circ(f_1) \neq \mathbf{type}^\circ(f_2) = \mathbf{type}^\circ(f_3)$ or $\mathbf{type}^\circ(f_1) = \mathbf{type}^\circ(f_2) \neq \mathbf{type}^\circ(f_3)$. Let e_1, e_2, e_3 be such that, $e_1^\circ = (f_2, f_3), e_2^\circ = (f_1, f_3), e_3^\circ = (f_1, f_2)$. Then the two cases correspond to $|\mathbf{type}^\circ(e_1)| = |\mathbf{type}^\circ(e_2)| = 2, |\mathbf{type}^\circ(e_3)| = 1$ and to $|\mathbf{type}^\circ(e_1)| = 1, |\mathbf{type}^\circ(e_2)| = |\mathbf{type}^\circ(e_3)| = 2$ respectively. Noticing that e_1, e_3 are both incoming or both outgoing edges of v completes the proof for the clockwise case. The proof for the counterclockwise case is formally identical. ◀

Proof of Lemma 11. For $v \in V(G)$, let $C_v \subseteq \mathbf{Exp}^\circ(G)$ be the expanded cycle. If it has a red vertex it is immediately enclosed by a unique red cycle R in $\mathbf{Exp}^\circ(G)$ by Lemma 9. Assuming C_v is not all red, it consists of alternating red subpaths and white subpaths. On contracting C_v we get a collection of clockwise red cycles outside sharing a common cut-vertex v . Notice that the new collection of red cycles consists of edges that R did not share with C_v . Also notice that (as a thought experiment) if we contracted the C_v 's that share a vertex with R , one at a time we would get an edge-disjoint set of red cycles with distinct cut vertices. Therefore, in $\mathbf{Col}^\circ(G)$, the red subgraph consists of a collection of connected components, each of which is a remnant of exactly one red cycle in $\mathbf{Exp}^\circ(G)$; these connected components consist of red cycles that touch externally at cut vertices. Hence both parts of the lemma follow. ◀

Proof of Lemma 22.

1. In a non-trivial strongly connected graph every vertex and edge lies on a cycle and therefore by Lemma 18 must be colored red or blue (or both).
2. Suppose there is a face f the boundary of which is not a directed cycle. Look at a directed dual (say clockwise) of the strongly connected component (just the component independently). This dual must be a DAG since the primal is strongly connected. The vertex f^* in the dual corresponding to face f of the strongly connected component has in degree at least one and out degree at least one since it has boundary edges of both orientations, hence the edges adjacent to f^* do not form a directed cut of the dual. Let o^* denote the dual vertex corresponding to the outer face of the SCC. In order to prove the claim, it is sufficient to show the existence of a directed cut C^* that separates f^* and o^* , since it would imply by cut cycle duality that there is a directed cycle C in the primal SCC that encloses the face f w.r.t the outer face and since the boundary of f is not a directed cycle, C must strictly enclose at least one edge of the boundary of f contradicting Lemma 14. To see the cut, consider a topological sort ordering of the dual (it is a DAG). Let the number of a dual vertex v^* in the ordering be denoted by $n(v^*)$. W.l.o.g, let $n(f^*) < n(o^*)$. Consider the partition of the dual vertices:

$$A = \{v^* \mid n(v^*) \leq n(f^*)\}, B = \{v^* \mid n(v^*) > n(f^*)\}$$

- By definition of topological sort, all edges across this partition must be directed from A to B , hence it is a directed cut, and therefore it must also contain a subset which is a minimal directed cut. But clearly the minimal cut is not the set of edges adjacent to f^* since it has both out and in degree at least one, hence proving the claim. Hence every face in the SCC of a sublayer must be a directed (hence colored) cycle (by Lemma 18).
3. Let H be an SCC of the sublayer. We observed from the proof above that no vertex in the dual of H , except possibly the vertex corresponding to the outer face of H , can have both in degree and out degree more than one. (i.e. every dual vertex, except the outer face is a source or a sink). Therefore if any dual vertex f^* has a directed path to o^* or vice versa, then the path must be an edge and we are done. Suppose there is no directed path from f^* to o^* and w.l.o.g. let f^* be a source. Consider the trivial directed cut C_1 :

$$A = \{f^*\}, B = V(H) \setminus A$$

This is a cut since there are no edges from B to A , and this cut clearly corresponds to the directed cycle which is the boundary of face f in H .

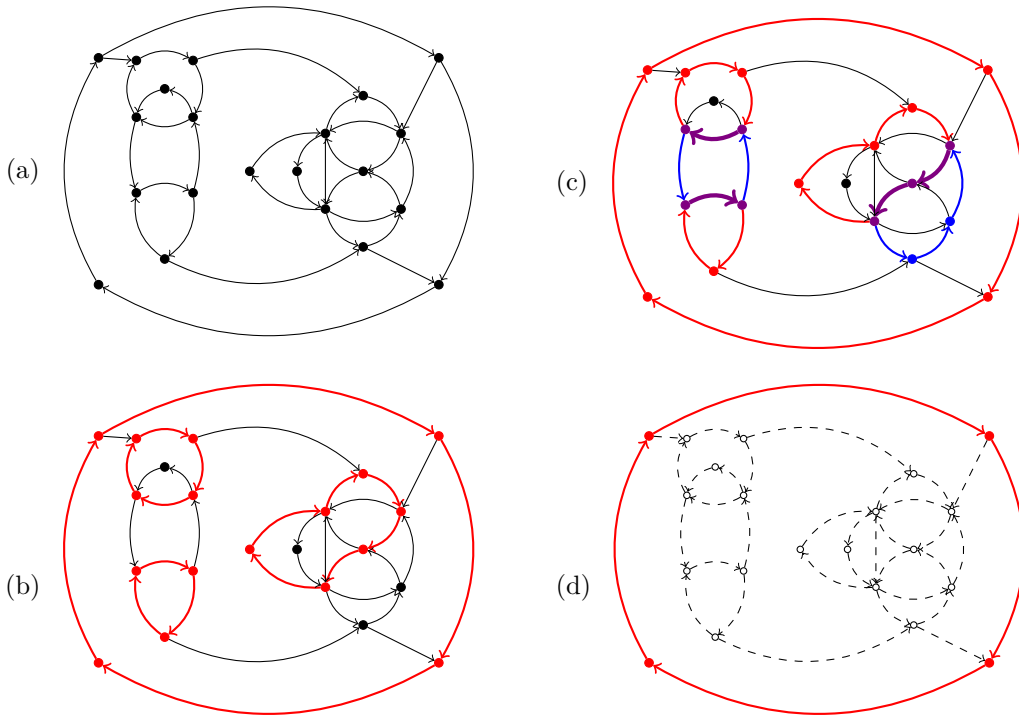
Now consider the cut C_2 :

$$A' = \{v^* \mid v^* \text{ is reachable from } f^*\}, B' = V(H) \setminus A'$$

Clearly this is a f^* - o^* cut with no edge from a vertex in A' to a vertex in B' and $o^* \in B'$. But this f^* - o^* cut is different from C_1 since f^* is a source vertex and hence A' has at least one more vertex than just f^* . Hence this corresponds to a directed cycle in H that strictly encloses at least some edge of f , and we again get a contradiction of Lemma 14. ◀

A.2 Details for Section 5

Proof. (of Lemma 29) Let e_i, e_{i+1} be two edges adjacent to v , that are also adjacent in the cyclic order of the drawing. Since they are adjacent in the drawing, they must enclose between them, a region, and hence a face, which is not the outer face. But the boundary of



■ **Figure 11** Figure (a) is a graph G . Figure (b) is the graph in (a) after labelling red edges using clockwise dual. We omit the cycle expansion and contraction procedure here.

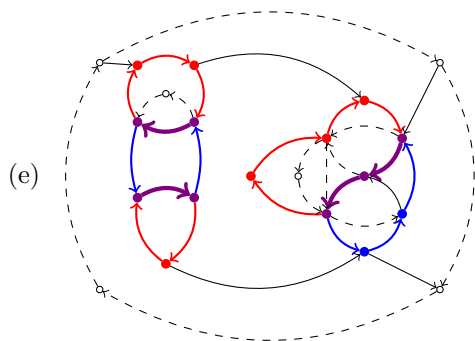
■ **Figure 12** Figure (c) shows G after applying blue labellings to each red layer we obtained in the previous figure. The vertices and edges colored purple are those that are red as well as blue. Figure (d) represents the sublayer (1,1). The dashed edges and empty vertices are not part of the layer.

every non-outer face in a mesh is a directed cycle, hence v, e_i, e_{i+1} lie on a directed cycle, with both edges adjacent to v . Hence one of them must be an out edge from v , and the other incident towards v . ◀

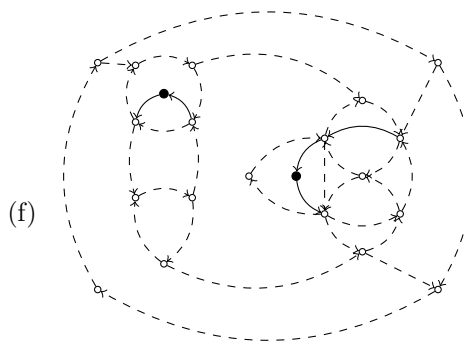
Proof of Proposition 31. Petals of two faces must be internally disjoint because the corresponding faces share the vertex v and two faces cannot have a non-contiguous intersection, by Lemma 21. ◀

Proof of Lemma 33.

1. Let x, y be the two vertices of attachment of the **2-bridge** B on $\mathbf{flower}(v)$. Since bridges are connected graphs without the edges of the corresponding cycle (by 3^{rd} property of definition 32), there must be an undirected path, p in the **bridge** connecting x, y , without using any edge of $\mathbf{flower}(v)$. If x and y were *not* on the same petal, then this path along with the other petals in $\mathbf{flower}(v)$, must clearly enclose a junction neighbour of v , say w (see Figure 4). Thus w is not adjacent to the outer face. Now since w is an internal junction vertex, and two of its adjacent faces are also adjacent to v , look at another face f adjacent to w and not adjacent to v . (Internal junction vertices have at least four adjacent faces.) The boundary of this face cannot touch B since that would make it a part of B and consequently w a vertex of attachment of B to $\mathbf{flower}(v)$. Therefore the boundary of f is enclosed within the paths p and the part of $\mathbf{flower}(v)$ that is also enclosed by p . Therefore f has no external edge, contradicting Lemma 22.



■ **Figure 13** Figure (e) figure represents the sublayer (2, 1).



■ **Figure 14** Figure (f) represents the sublayer (3, 1).

2. Let x_1, x_2, \dots, x_k be the vertices of attachment of the bridge B on $\mathbf{flower}(v)$, in the cyclic order of boundary of $\mathbf{flower}(v)$. Clearly if the vertices of attachment lie on more than two petals of v , then at least one petal will be completely enclosed by B , which is not possible since every petal must have at least one external edge. Lets say they lie on two adjacent petals, and the junction neighbour common to both of them is w . By the same argument as above, w must have an edge other than those of adjacent petals of v , that connect it to B . Therefore w must be a vertex of attachment of B to $\mathbf{flower}(v)$.
3. First we note that $\mathit{petal}_{w_i, u_j}$ will have an external vertex in it since the boundary of every face has at least one external vertex (Lemma 22), and segments (u_j, v) and (v, w_i) are internal. Let z be an external vertex on $\mathit{petal}_{w_i, u_j}$. The path p starts at external vertex r , comes to u_1, v, w_i , and reaches external vertex z on its way back to v . It will clearly divide \widetilde{M} into at least two parts by Jordan Curve theorem. Since p_{w_i, u_j} is just a wrap around the face f_{u_j, v, w_i} after z , is clear that since w_1, u_2, \dots, w_{i-1} and everything connected to them after removing p lie in one region, which we call V_{left} , and $w_{i+1}, u_{i+2}, \dots, w_d$ and everything connected to them after removing p lie in another disconnected region since p wraps around $f_{u, v, w}$. ◀

Proof of Lemma 36.

1. Suppose there were an undirected cycle in M of all internal segments, then this cycle must enclose a face whose boundaries are also all internal segments. This contradicts Lemma 22 as it states that every face must have at least one external edge, and hence segment. Hence there can be no cycle (directed or undirected) consisting of all internal segments, and consequently, no cycle (directed or undirected) of all internal vertices.
2. Let H be a 3-connected induced subgraph of M . By definition, $I(H)$ is obtained from M by removing all external edges and external non-junction vertices. Suppose $I(H)$ is not a tree, and hence consists of two or more disconnected trees. Let T_1 and T_2 be any two trees in $I(H)$. Let x be a vertex in T_1 and y be a vertex in T_2 . Since H is 3-connected, there must be at least three disjoint paths(undirected) between x and y . Clearly in a planar graph, if there are three disjoint paths between two vertices, one of the paths must be strictly enclosed in the closed region formed by other two. Therefore there must a path between x and y that is strictly enclosed inside the boundary of H , and hence does not contain any edge or vertex adjacent to the outer face of H . Hence x and y cannot become disconnected after removing external edges and external non-junction vertices leading to a contradiction that $I(H)$ is disconnected. Therefore $I(H)$ must be a tree. ◀

Proof of Lemma 39. Suppose $I'(P)$ has two junction vertices x and y that are sinks. They cannot be adjacent, so consider the unique undirected path in $I'(P)$ between x and y . There must be a source z on the path. Let neighbours of z be x', y' , lying on the path from x to z and from z to y respectively.

Let $\mathbf{B}_z^\circ(x')$ and $\mathbf{B}_z^\circ(y')$ denote the **bridges of flower**(z) with points of attachments x' and y' respectively. Then by the orientations of the edges we have: $|\widetilde{\mathbf{B}_z^\circ(x')}| \geq |\widetilde{\mathbf{B}_{x'}^\circ(z)}|$ which gives $|\widetilde{\mathbf{B}_z^\circ(x')}| > |\widetilde{\mathbf{B}_z^\circ(y')}|$ since $\mathbf{B}_z^\circ(y')$ is clearly a proper subgraph of $\mathbf{B}_{x'}^\circ(z)$ and $|\widetilde{\mathbf{B}_z^\circ(y')}| \geq |\widetilde{\mathbf{B}_{y'}^\circ(z)}|$ which gives $|\widetilde{\mathbf{B}_z^\circ(y')}| > |\widetilde{\mathbf{B}_z^\circ(x')}|$ which is clearly a contradiction. \blacktriangleleft

Proof of Lemma 42. A (weakly connected) component of V_{rem} is a bridge, attached to $petal_{w_i, u_i}$ or to $petal_{w_i, u_{i+1}}$ via its vertices of attachment. In the clique sum decomposition, these vertices of attachment will always contain a 1 or 2 separating clique, since if a bridge is attached to a petal via three or more nodes, the first and the last vertices of attachment form a separating pair that separates the bridge from **flower**(v). Hence it is a branch remaining in T_M after removing the 3 – *connected* piece node that is central in T_M . Since every branch after removal of the central piece of T_M has weight $\leq \frac{1}{2}(wt(M))$, every (weakly) connected component of V_{rem} has weight $\leq \frac{1}{2}(wt(M))$. \blacktriangleleft

A.3 Mesh Separator Algorithm

Proof of Lemma 43. We have the following two cases

1. For some i and $j \in \{i, i+1\}$, p_{w_i, u_j} , $wt(V_{rem}(w_i, u_j)) \geq \frac{1}{2}wt(M)$.

Then by Lemma 42, p_{w_i, u_j} separates $V_{rem}(w_i, u_j)$ from the rest of the graph, and also every weakly connected component in $V_{rem}(w_i, u_j)$ has weight $\leq \frac{1}{2}wt(M)$. Hence every weakly connected component in M after removing p_{w_i, u_j} has weight $\leq \frac{1}{2}wt(M)$.

2. For every p_{w_i, u_j} , $wt(V_{rem}(w_i, u_j)) \leq \frac{1}{2}wt(M)$.

We know that for any index i and $j \in \{i, i+1\}$, if $f = f_{u_j, v, w_i}$, then $wt(V_f) \leq wt(G)/12$ by the hypothesis of Lemma 28. Starting clockwise from p_{u_1, w_1} , at first V_{left} is small, and on shifting from p_{w_i, u_i} to $p_{w_i, u_{i+1}}$ or from $p_{w_i, u_{i+1}}$ to $p_{w_{i+1}, u_{i+1}}$, the increase in V_{left} is bounded above by $wt(V_f) + wt(V_{rem}(w_i, u_j)) + wt(\widetilde{\mathbf{B}_v^\circ(w_i)})$. Recall that

a. $wt(V_f) \leq wt(G)/12$ (by the hypothesis of Lemma 28).

b. $wt(\widetilde{V_{rem}(w_i, u_j)}) \leq \frac{1}{2}wt(M)$ (by hypothesis for this case).

c. $wt(\widetilde{\mathbf{B}_v^\circ(w_i)}) \leq \frac{1}{2}(wt(M) - wt(V_{rem}(w_i, u_j)))$ (by Lemma 40).

Thus the addition to V_{left} in each iteration is $\leq \frac{1}{12}wt(G) + wt(V_{rem}(w_i, u_j)) + \frac{1}{2}(wt(M) - \frac{1}{2}(wt(V_{rem}(w_i, u_j))))$, which is equal to $\frac{1}{12}wt(G) + \frac{1}{2}wt(V_{rem}(w_i, u_i)) + \frac{1}{2}(wt(M)) \leq \frac{1}{12}wt(G) + \frac{3}{4}wt(M)$. Thus we can stop the first time $wt(V_{left})$ is greater than $wt(G)/12$. So, we have $wt(V_{left}) \leq \frac{2}{12}wt(G) + \frac{3}{4}wt(M) \leq \frac{11}{12}wt(G)$, and $wt(V_{right}) \leq \frac{11}{12}wt(M)$, and $wt(V_f) \leq \frac{1}{12}wt(M)$, and $wt(v_{rem}) \leq \frac{1}{2}wt(M)$. Thus we have an upper bound of $\frac{11}{12}wt(G)$ on all the disconnected components. Hence p_{x_i, w_i} is a $\frac{11}{12}$ path separator. \blacktriangleleft

Order Reconfiguration Under Width Constraints

Emmanuel Arrighi ✉ 

University of Bergen, Norway

Henning Fernau ✉ 

University of Trier, Germany

Mateus de Oliveira Oliveira ✉ 

University of Bergen, Norway

Petra Wolf ✉ 

University of Trier, Germany

Abstract

In this work, we consider the following order reconfiguration problem: Given a graph G together with linear orders ω and ω' of the vertices of G , can one transform ω into ω' by a sequence of swaps of adjacent elements in such a way that at each time step the resulting linear order has cutwidth (pathwidth) at most k ? We show that this problem always has an affirmative answer when the input linear orders ω and ω' have cutwidth (pathwidth) at most $k/2$. Using this result, we establish a connection between two apparently unrelated problems: the reachability problem for two-letter string rewriting systems and the graph isomorphism problem for graphs of bounded cutwidth. This opens an avenue for the study of the famous graph isomorphism problem using techniques from term rewriting theory.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Mathematics of computing → Combinatorial optimization; Theory of computation → Equational logic and rewriting

Keywords and phrases Parameterized Complexity, Order Reconfiguration, String Rewriting Systems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.8

Funding *Emmanuel Arrighi*: Research Council of Norway (274526), IS-DAAD (309319).

Henning Fernau: DAAD PPP (57525246).

Mateus de Oliveira Oliveira: Research Council of Norway (288761), IS-DAAD (309319).

Petra Wolf: DFG project FE 560/9-1, DAAD PPP (57525246).

1 Introduction

In the field of reconfiguration, one is interested in studying relationships among solutions of a problem instance [17, 24, 27]. Here, by reconfiguration of one solution into another, we mean a sequence of steps where each step transforms a feasible solution into another. Three fundamental questions in this context are: (1) Is it the case that any two solutions can be reconfigured into each other? (2) Can any two solutions be reconfigured into each other in a polynomial number of steps? (3) Given two feasible solutions X and Y , can one find in polynomial time a reconfiguration sequence from X to Y ?

In this work, we study the reconfiguration problem in the context of linear arrangements of the vertices of a given graph G . The space of feasible solutions is the set of all linear orders of cutwidth (pathwidth) at most k for some given $k \in \mathbb{N}$. We say that a linear order ω can be reconfigured into a linear order ω' in width k if there is a sequence $\omega_1, \dots, \omega_m$ of linear orders of width at most k such that $\omega_1 = \omega$, $\omega_m = \omega'$ and for each $i \in \{2, \dots, m\}$, ω_i is obtained from ω_{i-1} by swapping two adjacent vertices. Our main result (Theorem 3) states that if ω and ω' are linear orders of cutwidth at most k , then ω can be reconfigured into ω' in width at most $2k$. Additionally, reconfiguration in width at most $2k$ can be done using at most $\mathcal{O}(n^2)$ swaps. Finally, a reconfiguration sequence can be found in polynomial time.



© Emmanuel Arrighi, Henning Fernau, Mateus de Oliveira Oliveira, and Petra Wolf; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 8; pp. 8:1–8:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Our results on reconfiguration of linear arrangements can be used to establish an interesting connection between two apparently unrelated computational problems: reachability for two-letter string rewriting and graph isomorphism.

A two-letter rewriting rule over a given alphabet Σ is a rewriting rule of the form $ab \rightarrow cd$ for letters $a, b, c, d \in \Sigma$. A two-letter string rewriting system is a collection R of two-letter string rewriting rules. The reachability problem for such a rewriting system R is the problem of determining whether a given string $x \in \Sigma^n$ can be transformed into a given string $y \in \Sigma^n$ by the application of a sequence of two-letter rewriting rules. On the other hand, in the graph isomorphism problem, we are given graphs G and G' and the goal is to determine whether there exists a bijection φ from the vertex set of G to the vertex set of G' in such a way that an edge $\{u, v\}$ belongs to G if and only if the edge $\{\varphi(u), \varphi(v)\}$ belongs to G' .

In order to describe more precisely the connections between two-letter term rewriting and graph isomorphism, we briefly discuss the notion of slices and unit decompositions. A *slice* is a graph \mathbf{S} where the vertices are partitioned into a center C and special in-frontier I and out-frontier O that can be used for composition. A slice \mathbf{S}_1 can be glued to a slice \mathbf{S}_2 if the out-frontier of \mathbf{S}_1 can be coherently matched with the in-frontier of \mathbf{S}_2 . In this case, the gluing gives rise to a bigger slice $\mathbf{S}_1 \circ \mathbf{S}_2$ which is obtained by matching the out-frontier of \mathbf{S}_1 with the in-frontier of \mathbf{S}_2 . A *unit slice* is a slice with a unique vertex in the center. Any slice \mathbf{S} can be decomposed into a sequence of unit slices. More specifically, a *unit decomposition* is a sequence $\mathbf{U} = \mathbf{S}_1 \mathbf{S}_2 \dots \mathbf{S}_n$ of unit slices with the property that for each $i \in [n - 1]$, \mathbf{S}_i can be glued to the slice \mathbf{S}_{i+1} . The result of gluing the unit slices in \mathbf{U} is a slice $\hat{\mathbf{U}}$ with n center vertices. Conversely, any slice \mathbf{S} with n center vertices can be written as a unit decomposition $\mathbf{U} = \mathbf{S}_1 \mathbf{S}_2 \dots \mathbf{S}_n$ with the property that $\hat{\mathbf{U}}$ is isomorphic to \mathbf{S} .

An important remark connecting unit decompositions and the notion of cutwidth is that if a slice \mathbf{S} has cutwidth k , then \mathbf{S} can be decomposed into a unit decomposition $\mathbf{U} = \mathbf{S}_1 \mathbf{S}_2 \dots \mathbf{S}_n$ where for each $i \in [n]$, \mathbf{S}_i has at most k vertices in each frontier. Therefore, if we let $\Sigma(k)$ denote the set of all unit slices with frontiers of size at most k , then any graph G with n vertices of cutwidth at most k can be written as a word (unit decomposition) of length n over the alphabet $\Sigma(k)$. In this work, for each $k \in \mathbb{N}$, we introduce a suitable two-letter string rewriting system $R(k)$ over the alphabet $\Sigma(k)$ with the following property: if \mathbf{U} and \mathbf{U}' are two unit decompositions over $\Sigma(k)$ and if \mathbf{U} can be transformed into \mathbf{U}' using the rewriting rules in $R(k)$, then the graphs $\hat{\mathbf{U}}$ and $\hat{\mathbf{U}}'$ are isomorphic. Our second main result is a partial converse for this property. More precisely, we show that given two unit decompositions \mathbf{U} and \mathbf{U}' over $\Sigma(k)$, if the graphs $\hat{\mathbf{U}}$ and $\hat{\mathbf{U}}'$ are isomorphic, then each of these unit decompositions can be transformed into one another by the application of rewriting rules from the string rewriting system $R(2k)$ (Theorem 11).

The proof of Theorem 11 is heavily based on Theorem 3. An important feature of this proof is that, given an isomorphism from $\hat{\mathbf{U}}$ to $\hat{\mathbf{U}}'$, one can construct a sequence of rewriting steps transforming \mathbf{U} into \mathbf{U}' . Conversely, given any such a sequence, we are able to construct an isomorphism from $\hat{\mathbf{U}}$ to $\hat{\mathbf{U}}'$. This result, together with the fact that unit decompositions of minimum cutwidth can be approximated in FPT time, implies that the graph isomorphism problem for graphs of cutwidth at most k is FPT-equivalent to the reachability problem for $R(2k)$ (Theorem 13).

Related Work. The reachability problem for a given string rewriting system R consists in determining whether a given string x can be transformed into a given string y by the application of rewriting rules from R . Reachability is a central problem in the field of string rewriting [6] and can also be studied under the light of term rewriting theory [19, 5, 1, 6].

The complexity of the reachability problem is highly dependent on the rewriting system R . For general rewriting systems, the problem becomes undecidable [6]. In the case of two-letter rewriting, reachability can be solved in PSPACE since in this case, strings never grow in size. It is also not difficult to design two-letter rewriting systems for which the reachability problem is PSPACE-complete. Nevertheless, our results imply that for each $k \in \mathbb{N}$, the $R(2 \cdot k)$ -reachability problem for unit decompositions of length n and width at most k is reducible to the graph isomorphism problem. Therefore, it can be solved in time $n^{\text{polylog}(n)}$, independently of k , using Babai's quasi-polynomial time algorithm for graph isomorphism [2]. An interesting question we leave unsolved is the complexity of $R(\alpha \cdot k)$ -reachability for unit decompositions of width at most k when α is a rational number with $1 \leq \alpha < 2$. In particular, we do not know if there is such an α for which the reachability problem becomes PSPACE-hard.

In the field of parameterized complexity theory [7, 6], a computational problem is said to be *fixed-parameter tractable (FPT)* with respect to a parameter k if it can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ on inputs of size n . Here $f: \mathbb{N} \rightarrow \mathbb{N}$ is a computable function depending only on the parameter k , but not on the size n of the input. The GRAPH ISOMORPHISM problem (GI for short) has been shown to be solvable in time $f(k) \cdot n^{\mathcal{O}(1)}$ (that is, FPT time) whenever the parameter k stands for eigenvalue multiplicity [3], treewidth [21], feedback vertex-set number [20], or size of the largest color class [9] of the involved graphs. On the other hand, GI can be solved in time $f_1(k) \cdot n^{f_2(k)}$ (that is, in XP time), whenever the parameter k stands for genus [23], rankwidth [15], maximum degree [22], size of an excluded topological subgraph [12], or size of an excluded minor [11]. We note that, in particular, Babai's algorithm and techniques have been recently used to improve the fastest FPT algorithm for graphs of treewidth at most k from $2^{\mathcal{O}(k^5 \cdot \log k)} \cdot n^{\mathcal{O}(1)}$ [21] to $2^{\mathcal{O}(k \cdot \text{polylog}(k))} \cdot n^{\mathcal{O}(1)}$ [14], and for graphs of maximum degree d , the fastest XP-algorithm has been improved from $n^{\mathcal{O}(d/\log d)}$ [4] to $n^{\text{polylog}(d)}$ [13]. In particular, it is worth noting that graphs of cutwidth k have maximum degree at most k and treewidth $\mathcal{O}(k)$. Therefore, isomorphism of graphs of cutwidth k can be solved in time $2^{\mathcal{O}(k \cdot \text{polylog}(k))} \cdot n^{\mathcal{O}(1)}$ [14]. This implies that $R(2 \cdot k)$ -reachability can be solved in $2^{\mathcal{O}(k \cdot \text{polylog}(k))} \cdot n^{\mathcal{O}(1)}$ time when restricted to unit decompositions of width at most k . Showing that isomorphism for graphs of cutwidth k can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ is still an open problem.

Another width parameter for linear orders that has been studied in the context of graph theory is the vertex separation number of a graph [8]. This parameter may be seen as an order theoretic interpretation of the notion of pathwidth. The techniques used to prove Theorem 3 can be generalized to prove that reconfiguration of linear orders of vertex separation number k can always be achieved in width at most $2 \cdot k$ (Theorem 16). While we do not provide a string-rewriting interpretation of this result, we do state it formally in Section 5 since this result may be of independent interest in the field of reconfiguration.

2 Preliminaries

Basics. We let \mathbb{N} denote the set of natural numbers, including 0, and \mathbb{N}_+ denote the set of positive natural numbers. For each $n \in \mathbb{N}_+$, we let $[n] = \{1, \dots, n\}$. As a degenerate case, we let $[0] = \emptyset$. Given a finite set S , we let $\mathcal{P}(S)$ be the set of all subsets of S . For each $k \in \mathbb{N}$, we let $\mathcal{P}(S, k)$ and $\mathcal{P}(S, \leq k)$ be the sets of subsets of S of size exactly k and at most k , respectively.

8:4 Order Reconfiguration Under Width Constraints

Graphs. In this work, graphs are simple and undirected. Given a graph G we let $V(G)$ denote the vertex set of G and $E(G)$ denote the edge set of G . Given a subset $S \subseteq V(G)$, we let $G[S]$ be the subgraph of G induced by S . More precisely, $V(G[S]) = S$ and $E(G[S]) = E(G) \cap \mathcal{P}(S, 2)$. An *isomorphism* from a graph G to a graph G' is a bijection $\varphi : V(G) \rightarrow V(G')$ such that for each $v, u \in V(G)$, $\{v, u\} \in E(G)$ if and only if $\{\varphi(v), \varphi(u)\} \in E(G')$. If such an isomorphism exists, we say that G is *isomorphic* to G' .

Order. Let V be a set with $|V| = n$. A linear order on V is a bijection $\omega : [n] \rightarrow V$. Intuitively, for each $j \in [n]$ and $v \in V$, $\omega(j) = v$ indicates that v is the j -th element of ω . If $S \subseteq [n]$, then we let $\omega(S) = \{\omega(j) : j \in S\}$ be the image of S under ω . Given linear orders $\omega, \omega' : [n] \rightarrow V$ of V and a number $i \in [n-1]$, we write $\omega \xrightarrow{i} \omega'$ to indicate that ω' is obtained from ω by swapping the order of the vertices at positions i and $i+1$. More precisely, $\omega'(j) = \omega(j)$ for every $j \in [n] \setminus \{i, i+1\}$, $\omega'(i) = \omega(i+1)$, and $\omega'(i+1) = \omega(i)$.

Let $\omega : [n] \rightarrow V$ be a linear order on a set V . Let $S \subseteq V$. We let $\omega^S : [|S|] \rightarrow S$ be the linear order induced by ω on S . More precisely, if we write the elements of S in increasing order according to ω , then for each $i \in [|S|]$, $\omega^S(i)$ is the i -th element in this sequence.

Order Reconfiguration. We say that ω can be reconfigured into ω' in one swap, and denote this fact by $\omega \rightarrow \omega'$, if there exists some $i \in [n]$ such that $\omega \xrightarrow{i} \omega'$. We say that ω can be reconfigured into ω' in at most r swaps, and denote this fact by $\omega \rightarrow_r \omega'$, if there are numbers $r' \in [r]$, $i_1, \dots, i_{r'} \in [n]$, and linear orders $\omega_0, \dots, \omega_{r'}$ such that

$$\omega = \omega_0 \xrightarrow{i_1} \omega_1 \xrightarrow{i_2} \dots \xrightarrow{i_{r'}} \omega_{r'} = \omega'.$$

We call this sequence a *reconfiguration sequence* from ω to ω' . The mere existence of a (possibly empty) reconfiguration sequence from ω to ω' is also written as $\omega \rightarrow^* \omega'$.

Composition of Linear Orders. Let $i \in \{0, \dots, n\}$, and $\omega, \omega' : [n] \rightarrow V$. We let $\omega \oplus_i \omega' : [n] \rightarrow V$ be the linear order that orders the vertices in the subset $\omega([i]) \subseteq V$ according to ω followed by the vertices in the subset $V \setminus \omega([i])$, ordered according to ω' . More precisely, $\omega \oplus_i \omega'$ is defined as follows for each $j \in [n]$.

$$\omega \oplus_i \omega'(j) = \begin{cases} \omega(j) & \text{if } j \leq i, \\ \omega'^{V \setminus \omega([i])}(j-i) & \text{if } j > i. \end{cases} \quad (1)$$

We note that in particular, $\omega \oplus_0 \omega' = \omega'$ and $\omega \oplus_n \omega' = \omega$.

String Rewriting. A two-letter string rewriting systems is a pair (Σ, R) where Σ is a finite, non-empty set of symbols (an alphabet), and $R \subseteq \Sigma^2 \times \Sigma^2$ is a set of rewriting rules of the form $ab \rightarrow cd$. Let x and y be strings in Σ^n and $i \in [n-1]$. We say that x can be transformed into y by applying a rewriting rule $ab \rightarrow cd$ at position i if $x_i x_{i+1} = ab$, $y_i y_{i+1} = cd$ and $x_j = y_j$ for $j \notin \{i, i+1\}$. We write $x \xrightarrow{i} y$ to denote that x can be transformed into y by the application of some rewriting rule at position i . We write $x \rightarrow y$ to denote that x can be transformed into y by the application of some rewriting rule at some position $i \in [n-1]$. We say that y is reachable from x if there is a sequence of strings $x = x_0, x_1, \dots, x_m = y$ such that $x_{i-1} \rightarrow x_i$ for each $i \in [m]$. We write $x \rightarrow_* y$ to denote that y is reachable from x . We say that x and y are R -equivalent if $x \rightarrow_* y$ and $y \rightarrow_* x$.

3 Linear Order Reconfiguration

Let G be an n -vertex graph. Given sets $S, S' \subseteq V(G)$, we let $E(G, S, S') = \{\{u, v\} \in E(G) : u \in S, v \in S'\}$ be the set of edges with one endpoint in S and the other endpoint in S' . As a special case, we define $E(G, S) = E(G, S, V(G) \setminus S)$. We will often make use of the following monotonicity property without explicit mentioning: If $T \subseteq S$ and $T' \subseteq S'$, then $|E(G, T, T')| \leq |E(G, S, S')|$.

► **Definition 1** (Cutwidth). *Let G be an n -vertex undirected graph. Let $\omega : [n] \rightarrow V(G)$ be a linear order on the vertices of G . For each $p \in [n]$, we let $\text{cw}(G, \omega, p) = |E(G, \omega([p-1]))|$ be the number of edges that have one endpoint in the first $p-1$ vertices of the linear order ω and the other endpoint in the remaining vertices. The cutwidth of the linear order ω is defined as $\text{cw}(G, \omega) = \max_{p \in [n]} \text{cw}(G, \omega, p)$. The cutwidth of the graph G is defined as $\text{cw}(G) = \min_{\omega} \text{cw}(G, \omega)$, where ω ranges over all linear orders on the vertex set $V(G)$.*

For each $k \in \mathbb{N}$, and each n -vertex graph G , we let $\text{CW}(G, k) = \{\omega : [n] \rightarrow V(G) : \text{cw}(G, \omega) \leq k\}$ be the set of linear orders of $V(G)$ of cutwidth at most k . We say that ω can be *reconfigured* into ω' in cutwidth at most k if there is a *reconfiguration sequence*

$$\omega = \omega_0 \xrightarrow{i_1} \omega_1 \xrightarrow{i_2} \dots \xrightarrow{i_r} \omega_r = \omega'$$

such that for each $j \in \{0, \dots, r\}$, $\omega_j \in \text{CW}(G, k)$.

► **Problem 2** (Bounded Cutwidth Order Reconfiguration). *Let G be an n -vertex graph, $\omega, \omega' : [n] \rightarrow V(G)$ be linear orders on the vertex set of G , and $k \in \mathbb{N}$. Is it true that ω can be reconfigured into ω' in cutwidth at most k ?*

It should be clear that if k is smaller than the cutwidth of the graph G , then the answer for Problem 2 is trivially NO since in this case neither ω nor ω' are in $\text{CW}(G, k)$. On the other hand, we will show in Theorem 3 below that the answer is always YES if k is at least twice the cutwidth of the thickest input linear order.

► **Theorem 3.** *Let G be an n -vertex graph and $\omega, \omega' : [n] \rightarrow V(G)$ be linear orders of $V(G)$ of cutwidth at most k . Then, ω can be reconfigured into ω' in cutwidth at most $\text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$.*

To prove this theorem, we need the following three lemmas.

► **Lemma 4.** *Let G be an n -vertex graph, $S \subseteq V(G)$ and $\omega : [n] \rightarrow V(G)$ be a linear order on $V(G)$. Then, ω^S is a linear order on $V(G[S])$. Additionally, $\text{cw}(G[S], \omega^S) \leq \text{cw}(G, \omega)$.*

Proof. As $S = V(G[S])$, ω^S is a linear order on $V(G[S])$. Let $p \in [|S|]$ and let $p' \in [n]$ be the unique number such that $\omega^S(p) = \omega(p')$. Then,

$$\begin{aligned} \text{cw}(G[S], \omega^S, p) &= |E(G[S], \omega^S([p-1]))| \\ &= |E(G[S], \omega^S([p-1]), \{\omega^S(r) : r \geq p\})| \\ &= |E(G, \omega^S([p-1]), \{\omega^S(r) : r \geq p\})| \\ &\leq |E(G, \omega([p'-1]))| \\ &= \text{cw}(G, \omega, p') \\ &\leq \text{cw}(G, \omega), \end{aligned}$$

as $\omega^S([p-1]) \subseteq \omega([p'-1])$ and $\{\omega^S(r) : r \geq p\} \subseteq \{\omega(r') : r' \geq p'\} = V(G) \setminus \omega([p'-1])$. ◀

8:6 Order Reconfiguration Under Width Constraints

► **Lemma 5.** *Let G be an n -vertex graph and $\omega, \omega' : [n] \rightarrow V(G)$ be linear orders of $V(G)$ with cutwidth of at most k . Then, for each $i \in [n]$, $\omega \oplus_i \omega'$ has cutwidth at most $\text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$.*

Proof. Let $i, p \in [n]$. There are two cases to be analyzed. By definition, we have that

$$\text{cw}(G, \omega \oplus_i \omega', p) = |E(G, \omega \oplus_i \omega'([p-1]))| = |E(G, \omega \oplus_i \omega'([p-1]), V(G) \setminus \omega \oplus_i \omega'([p-1]))|.$$

First, if $p \leq i$, then we have

$$\text{cw}(G, \omega \oplus_i \omega', p) = |E(G, \omega([p-1]), V(G) \setminus \omega([p-1]))| = \text{cw}(G, \omega, p) \leq \text{cw}(G, \omega).$$

Secondly, if $p > i$, then we have

$$\begin{aligned} \text{cw}(G, \omega \oplus_i \omega', p) &= |E(G, \omega \oplus_i \omega'([p-1]), V(G) \setminus \omega \oplus_i \omega'([p-1]))| \\ &\stackrel{(a)}{=} |E(G, (\omega \oplus_i \omega'([p-1])) \cap \omega([i]), V(G) \setminus \omega \oplus_i \omega'([p-1]))| \\ &\quad + |E(G, (\omega \oplus_i \omega'([p-1])) \setminus \omega([i]), V(G) \setminus \omega \oplus_i \omega'([p-1]))| \\ &\stackrel{(b)}{\leq} \text{cw}(G, \omega, i+1) + \text{cw}(G[V(G) \setminus \omega([i]), \omega'^{V(G) \setminus \omega([i])}], p-i) \\ &\leq \text{cw}(G, \omega) + \text{cw}(G, \omega'). \end{aligned}$$

For Equality (a), observe that $\{(\omega \oplus_i \omega'([p-1])) \cap \omega([i]), (\omega \oplus_i \omega'([p-1])) \setminus \omega([i])\}$ is a partition of $\omega \oplus_i \omega'([p-1])$. To understand Inequality (b), we need two arguments. As $\omega([i]) \subseteq \omega \oplus_i \omega'([p-1])$,

$$E(G, (\omega \oplus_i \omega'([p-1])) \cap \omega([i]), V(G) \setminus (\omega \oplus_i \omega'([p-1]))) \subseteq E(G, \omega([i]), V(G) \setminus \omega([i])),$$

which shows that the cardinality of the first set is upper-bounded by $\text{cw}(G, \omega, i+1)$. As the edges in $E(G, (\omega \oplus_i \omega'([p-1])) \setminus \omega([i]), V(G) \setminus (\omega \oplus_i \omega'([p-1])))$ only connect vertices with positions beyond i within $\omega \oplus_i \omega'$, after an index shift, we see that only the linear order ω' really matters, which explains the inequality

$$|E(G, (\omega \oplus_i \omega'([p-1])) \setminus \omega([i]), V(G) \setminus (\omega \oplus_i \omega'([p-1])))| \leq \text{cw}(G[V(G) \setminus \omega([i]), \omega'^{V(G) \setminus \omega([i])}], p-i).$$

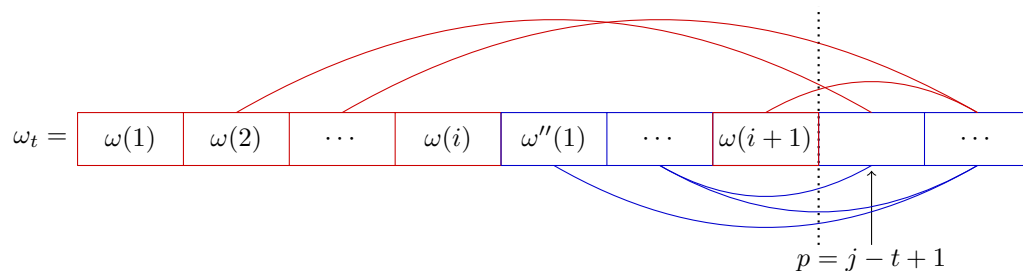
For the last inequality, apply Lemma 4 to derive $\text{cw}(G[V(G) \setminus \omega([i]), \omega'^{V(G) \setminus \omega([i])}]) \leq \text{cw}(G, \omega')$. As p is arbitrary, $\text{cw}(G, \omega \oplus_i \omega') \leq \text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$ follows for each $i \in [n]$. ◀

► **Lemma 6.** *Let G be an n -vertex graph, $\omega, \omega' : [n] \rightarrow V(G)$ be linear orders on $V(G)$ of cutwidth at most k and $i \in \{0, \dots, n-1\}$ be an integer. Then, $\omega \oplus_i \omega'$ can be reconfigured into $\omega \oplus_{i+1} \omega'$ in cutwidth at most $\text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$.*

Proof. By Lemma 5, $\omega \oplus_i \omega'$ and $\omega \oplus_{i+1} \omega'$ have cutwidth at most $\text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$. Let $j \in [n]$ such that $\omega \oplus_i \omega'(j) = \omega(i+1)$, i.e., j is the position of $\omega(i+1)$ in $\omega \oplus_i \omega'$. As for each $p \in [i]$, $\omega \oplus_i \omega'(p) = \omega \oplus_{i+1} \omega'(p) = \omega(p)$, we have $j > i$. Let us consider the following sequence of swaps:

$$\omega \oplus_i \omega' = \omega_0 \xrightarrow{j-1} \omega_1 \xrightarrow{j-2} \dots \xrightarrow{i+1} \omega_{j-i-1} = \omega \oplus_{i+1} \omega'.$$

If $j = i+1$, this sequence is empty and $\omega \oplus_i \omega' = \omega \oplus_{i+1} \omega'$. At each step of this sequence, we swap $\omega(i+1)$ with its left neighbor. This brings $\omega(i+1)$ from position j to position $i+1$. By doing this, we transform $\omega \oplus_i \omega'$ into $\omega \oplus_{i+1} \omega'$.



■ **Figure 1** Illustration of the key part in Lemma 6. In this figure $\omega'' = \omega^{V \setminus \omega([i+1])}$. The red part of the linear order follows the linear order ω for the first $i+1$ elements, and the blue part of the linear order follows ω' for the remaining elements. Then, the set of edges crossing the cut at position $p = j - t + 1$ can be split in two, the set of edges that start from the red part and the set of edge that start from the green part. The number of red edges is bounded by the cutwidth of ω and the number of green edges is bounded by the cutwidth of ω' .

Inductively, we show that each element ω_t in the sequence has cutwidth upper-bounded by $\text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$. By Lemma 5, $\text{cw}(G, \omega_0) = \text{cw}(G, \omega \oplus_i \omega') \leq \text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$, which proves the induction basis. Let $t \in [j-i-1]$ and $p \in [n]$ be two integers. As induction hypothesis, we have $\text{cw}(G, \omega_{t-1}) \leq \text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$. If $p \leq j-t$ or $p > j-t+1$, then we have $\omega_{t-1}([p-1]) = \omega_t([p-1])$, so $\text{cw}(G, \omega_t, p) = \text{cw}(G, \omega_{t-1}, p) \leq \text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$ by induction hypothesis. Otherwise, $p = j-t+1 \in \{i, \dots, j\}$ (Figure 1) and we have

$$\begin{aligned} \text{cw}(G, \omega_t, p) &= |E(G, \omega_t([p-1]))| \\ &= |E(G, \omega_t([p-1]), \{\omega_t(r) : r \geq p\})| \\ &= |E(G, \omega_t([i] \cup \{p-1\}), \{\omega_t(r) : r \geq p\})| \\ &\quad + |E(G, \{\omega_t(l) : i < l < p-1\}, \{\omega_t(r) : r \geq p\})|. \end{aligned}$$

By definition of ω_t and p , we have $\omega_t(p-1) = \omega_t(j-t) = \omega(i+1)$. Therefore, we have $|E(G, \omega_t([i] \cup \{p-1\}), \{\omega_t(r) : r \geq p\})| = |E(G, \omega([i+1]), \{\omega_t(r) : r \geq j-t+1\})|$. As we are swapping $\omega(i+1)$ leftwards, $\{\omega_t(r) : r \geq j-t+1\} \subseteq \{\omega(r) : r \geq i+2\} = V(G) \setminus \omega([i+1])$. Again by definition of ω_t and p , the elements in $\{\omega_t(l) : i < l < p-1\}$ are ordered according to ω' , which is also true for $\{\omega_t(r) : r \geq p\}$. More formally, $\{\omega_t(l) : i < l < p-1\} = \{\omega \oplus_i \omega'(l) : i+1 \leq l \leq p-2\} = \{\omega^{V(G) \setminus \omega([i+1])}(l') : l' \leq p-2-i\}$ and $\{\omega_t(r) : r \geq p\} = \{\omega \oplus_i \omega'(r) : r \geq p\} = \{\omega^{V(G) \setminus \omega([i+1])}(r') : r' \geq p-i-1\}$. Therefore,

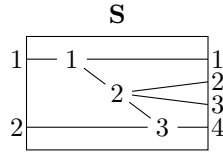
$$\begin{aligned} \text{cw}(G, \omega_t, p) &\leq \text{cw}(G, \omega, i+2) + \text{cw}(G[V(G) \setminus \omega([i+1])], \omega^{V(G) \setminus \omega([i+1])}, p-i-1) \\ &\leq \text{cw}(G, \omega) + \text{cw}(G[V(G) \setminus \omega([i+1])], \omega^{V(G) \setminus \omega([i+1])}) \\ &\leq \text{cw}(G, \omega) + \text{cw}(G, \omega') \\ &\leq 2k. \end{aligned}$$

To achieve the penultimate inequality, we again apply Lemma 4. ◀

Proof of Theorem 3. Consider the following sequence: $\omega = \omega' \oplus_0 \omega \rightarrow^* \omega' \oplus_1 \omega \rightarrow^* \dots \rightarrow^* \omega' \oplus_n \omega = \omega'$. By Lemma 6, one can realize each step in cutwidth at most $\text{cw}(G, \omega) + \text{cw}(G, \omega') \leq 2k$, which then also upper-bounds the whole reconfiguration sequence. ◀

4 Slice Rewriting System

Slices. Let $\mathcal{I} = \{[a] : a \in \mathbb{N}\}$ denote the set of intervals of the form $[a] = \{1, \dots, a\}$ for $a \in \mathbb{N}$ (recall that $[0] = \emptyset$). We let $\mathcal{I}_0 = \{\{0\} \times [a] : [a] \in \mathcal{I}\}$, and $\mathcal{I}_1 = \{\{1\} \times [a] : [a] \in \mathcal{I}\}$. A *slice* $\mathbf{S} = (I, C, O, E)$ is a (multi-)graph where the vertex set $V = I \dot{\cup} C \dot{\cup} O$ is partitioned into an *in-frontier* $I \in \mathcal{I}_0$, a *center* $C \in \mathcal{I}$ and an *out-frontier* $O \in \mathcal{I}_1$, and E is a multiset of unordered pairs from $I \cup C \cup O$ in such a way that vertices of $I \cup O$ have degree exactly 1, there is no edge between any two vertices in I , and no edge between any two edges in O . We depict slices as in Figure 2. We define slices using multigraphs, as the gluing operation, defined below, can take slices which are simple graphs, and create a slice which is a multigraph (see Figure 5). Given a slice \mathbf{S} , we define $I(\mathbf{S})$ as the in-frontier of \mathbf{S} , $O(\mathbf{S})$ as the out-frontier of \mathbf{S} , and $C(\mathbf{S})$ as the center vertices of \mathbf{S} . The *width* of a slice \mathbf{S} is defined as $\mathbf{w}(\mathbf{S}) = \max(|I(\mathbf{S})|, |O(\mathbf{S})|)$.



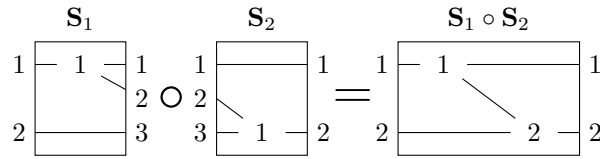
■ **Figure 2** Slices are drawn as tiles. This figure depicts the slice $\mathbf{S} = (I, C, O, E)$ where $I = \{(0, 1), (0, 2)\}$, $C = \{1, 2, 3\}$, $O = \{(1, 1), (1, 2), (1, 3), (1, 4)\}$ and $E = \{\{(0, 1), 1\}, \{(0, 2), 3\}, \{1, 2\}, \{2, 3\}, \{1, (1, 1)\}, \{2, (1, 2)\}, \{2, (1, 3)\}, \{3, (1, 4)\}\}$. We omit the first element of the pair for frontier vertices and use the following convention. The in-frontier vertices are on the left of the tile and the out-frontier vertices are on the right of the tile. If the frontier vertices are not explicitly mentioned in the drawing, we assume that frontier vertices are ordered from top to bottom as in this drawing.

Gluing Slices. A slice $\mathbf{S}_1 = (I_1, C_1, O_1, E_1)$ can be glued to $\mathbf{S}_2 = (I_2, C_2, O_2, E_2)$ if for some interval $[a] \in \mathcal{I}$, $O_1 = \{1\} \times [a]$ and $I_2 = \{0\} \times [a]$. In this case, the gluing gives rise to the slice $\mathbf{S}_1 \circ \mathbf{S}_2 = (I_1, C_1 \cup (|C_1| \oplus C_2), O_2, E)$ where $|C_1| \oplus C_2 = \{|C_1| + 1, |C_1| + 2, \dots, |C_1| + |C_2|\}$,

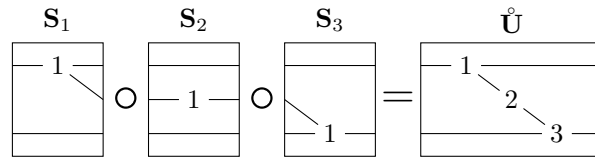
$$\begin{aligned}
 E = & \{\{x, y\} \in E_1 : x, y \in I_1 \cup C_1\} \\
 & \cup \{\{x, y\} \in E_2 : x, y \in O_2\} \\
 & \cup \{\{x, y + |C_1|\} : \{x, y\} \in E_2 \wedge x \in O_2 \wedge y \in C_2\} \\
 & \cup \{\{x + |C_1|, y + |C_1|\} : \{x, y\} \in E_2 \wedge x, y \in C_2\} \\
 & \cup \{\{x, y\} : \exists i, \{x, (1, i)\} \in E_1 \wedge y \in O_2 \wedge \{(0, i), y\} \in E_2\} \\
 & \cup \{\{x, y\} : \exists i, \{x, (1, i)\} \in E_1 \wedge y \in |C_1| \oplus C_2 \wedge \{(0, i), y - |C_1|\} \in E_2\}.
 \end{aligned}$$

Note that the gluing operation is associative. Therefore, we will not write parentheses for the gluing of more than two slices. Figure 3 illustrates the gluing of two slices.

Unit Slices and Unit Decompositions. We say that a slice is a *unit slice* if it has a unique vertex in its center. A *unit decomposition* is a sequence $\mathbf{U} = \mathbf{S}_1 \mathbf{S}_2 \dots \mathbf{S}_n$, where \mathbf{S}_i are unit slices and $\mathbf{S}_i \circ \mathbf{S}_{i+1}$ is well defined for each $i \in [n - 1]$. The *slice associated to a unit decomposition* \mathbf{U} is defined as $\mathring{\mathbf{U}} = \mathbf{S}_1 \circ \mathbf{S}_2 \circ \dots \circ \mathbf{S}_n$ (Figure 4). Note that if the in-frontier of \mathbf{S}_1 and the out-frontier of \mathbf{S}_n are empty, then $\mathring{\mathbf{U}}$ is just a multigraph with vertex set $[n]$ (Figure 5). For each $k \in \mathbb{N}$, we define the alphabet $\Sigma(k)$ as the set of all unit slices of width at most k . We let $\Sigma(k)^\circledast$ denote the set of all unit decompositions over $\Sigma(k)$.



■ **Figure 3** Gluing of two slices \mathbf{S}_1 and \mathbf{S}_2 . The gluing operation is a way to merge two slices into one. In this example, the edge from the center vertex 1 from \mathbf{S}_1 to the out-frontier vertex $(1, 2)$ is stitched to the edge from the in-frontier vertex $(0, 2)$ to the center vertex 1 from \mathbf{S}_2 to form the edge between the center vertices 1 and 2 in $\mathbf{S}_1 \circ \mathbf{S}_2$. The stitching of edges is done following the order of the frontier vertices.



■ **Figure 4** Slice associated to the unit decomposition $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2\mathbf{S}_3$. The gluing operation is associative, therefore parentheses are not needed.

The order of the unit slices in a unit decomposition $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2 \dots \mathbf{S}_n$ induces a linear order $\omega_{\mathbf{U}}$ on the vertices of the slice $\mathring{\mathbf{U}}$. This linear order sets $\omega_{\mathbf{U}}(i) = (0, i)$ for each $(0, i) \in I(\mathbf{S}_1)$, $\omega_{\mathbf{U}}(|I(\mathbf{S}_1)| + i) = i$ for each $i \in \{1, \dots, n\}$ and $\omega_{\mathbf{U}}(|I(\mathbf{S}_1)| + n + i) = (1, i)$ for each $(1, i) \in O(\mathbf{S}_n)$.

Given a unit decomposition $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2 \dots \mathbf{S}_n$ in $\Sigma(k)^\circledast$, we let $\mathbf{w}(\mathbf{U}) = \max_{i \in [n]} \mathbf{w}(\mathbf{S}_i)$.

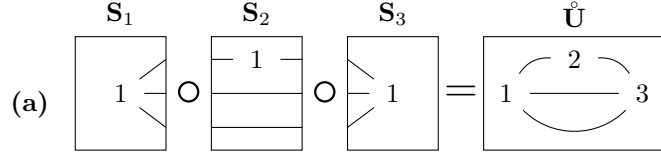
► **Proposition 7.** *Let $k \in \mathbb{N}$, and $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2 \dots \mathbf{S}_n$ be a unit decomposition in $\Sigma(k)^\circledast$, and $\omega_{\mathbf{U}}$ be the linear order induced by \mathbf{U} on $\mathring{\mathbf{U}}$. Then, $\text{cw}(\mathring{\mathbf{U}}, \omega_{\mathbf{U}}) = \mathbf{w}(\mathbf{U})$.*

Proof. This follows by noticing that for each vertex p in $\{1, \dots, |I(\mathbf{S}_1)|\}$, $\text{cw}(\mathring{\mathbf{U}}, \omega_{\mathbf{U}}, p) \leq |I(\mathbf{S}_1)|$, for each p in $\{|I(\mathbf{S}_1)| + n, \dots, |I(\mathbf{S}_1)| + n + |O(\mathbf{S}_n)|\}$, $\text{cw}(\mathring{\mathbf{U}}, \omega_{\mathbf{U}}, p) \leq O(\mathbf{S}_n)$, and for each $p \in \{|I(\mathbf{S}_1)| + 1, \dots, |I(\mathbf{S}_1)| + n\}$, $\text{cw}(\mathring{\mathbf{U}}, \omega_{\mathbf{U}}, p) \leq \mathbf{w}(\mathbf{S}_{i-|I(\mathbf{S}_1)|})$. ◀

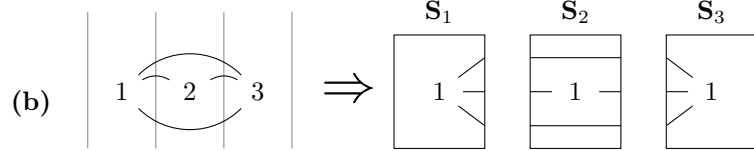
► **Proposition 8.** *Let G be an n -vertex graph and ω be a linear order on the vertices of G of cutwidth k . Then, we can construct in time $\mathcal{O}(kn)$ a unit decomposition \mathbf{U} such that $\omega = \omega_{\mathbf{U}}$.*

Proof. We will do this construction by first drawing the graph G in the plane. G does not need to be planar for this construction to work. First, we will place the vertices of G on a straight line L isomorphic to \mathbb{R} . The i th vertex of G with respect to the linear order ω is placed at the coordinate i on the line. Then edges are drawn as curves between their endpoints. Now, we will draw $n + 1$ lines perpendicular to L at coordinates $\{-0.5, 0.5, 1.5, \dots, n - 0.5, n + 0.5\}$. We call these lines cut-lines. The cutwidth of ω is k , therefore each cut-line intersects at most k edges in the drawing of G . We put a vertex at the intersection of a cut-line and an edge. The graph between two consecutive cut-lines defines a unit slice of width at most k . Taking all those slices in the order induced by ω on the line L gives a unit decomposition \mathbf{U} of width k such that $\omega = \omega_{\mathbf{U}}$. Figure 6 illustrates this construction. ◀

Equivalence of Slices. Let $\mathbf{S}_1 = (I_1, C_1, O_1, E_1)$ and $\mathbf{S}_2 = (I_2, C_2, O_2, E_2)$ be two slices. We say that \mathbf{S}_1 is *equivalent* to \mathbf{S}_2 , denoted by $\mathbf{S}_1 \sim \mathbf{S}_2$, if and only if $I_1 = I_2$, $O_1 = O_2$, $C_1 = C_2$, and there is an isomorphism ϕ from \mathbf{S}_1 to \mathbf{S}_2 such that the restriction of ϕ to I_1 and O_1 is the identity function. In other words, \mathbf{S}_1 and \mathbf{S}_2 are equivalent if they are equal up to the renaming of the center vertices.

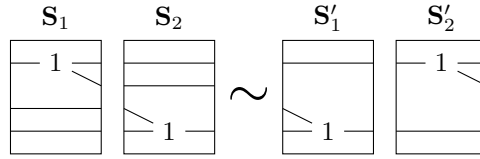


■ **Figure 5** Slice associated to the unit decomposition $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2\mathbf{S}_3$. The resulting slice does not have any vertex in its frontier. It can therefore be seen as a multigraph on 3 vertices.



■ **Figure 6** Slicing of the graph G on the left into a unit decomposition \mathbf{U} on the right.

We let $\mathcal{R}(k) \subseteq \Sigma(k)^2 \times \Sigma(k)^2$ be the set of all rewriting rules of the form $\mathbf{S}_1\mathbf{S}_2 \rightarrow \mathbf{S}'_1\mathbf{S}'_2$ such that $\mathbf{S}_1 \circ \mathbf{S}_2 \sim \mathbf{S}'_1 \circ \mathbf{S}'_2$. Call two unit decompositions $\mathbf{U}, \mathbf{U}' \in \Sigma(k)^\otimes$ *locally $\mathcal{R}(k)$ -equivalent*, and denote this fact by $\mathbf{U} \stackrel{k}{\sim} \mathbf{U}'$, if there exist $\mathbf{W}, \mathbf{W}' \in \Sigma(k)^\otimes$ and $\mathbf{S}_1, \mathbf{S}'_1, \mathbf{S}_2, \mathbf{S}'_2 \in \Sigma(k)$ with $\mathbf{S}_1 \circ \mathbf{S}_2 \sim \mathbf{S}'_1 \circ \mathbf{S}'_2$ such that $\mathbf{U} = \mathbf{W}\mathbf{S}_1\mathbf{S}_2\mathbf{W}'$ and $\mathbf{U}' = \mathbf{W}\mathbf{S}'_1\mathbf{S}'_2\mathbf{W}'$ (Figure 7).



■ **Figure 7** Local Equivalence. $\mathbf{S}_1\mathbf{S}_2$ is (locally) $\mathcal{R}(4)$ -equivalent to $\mathbf{S}'_1\mathbf{S}'_2$.

We let $\stackrel{k}{\equiv} \subseteq \Sigma(k)^\otimes \times \Sigma(k)^\otimes$ be the equivalence relation defined on unit decompositions by taking the reflexive, symmetric and transitive closure of $\stackrel{k}{\sim}$. If $\mathbf{U} \stackrel{k}{\equiv} \mathbf{U}'$, then we say that \mathbf{U}' is $\mathcal{R}(k)$ -equivalent to \mathbf{U} . We note that there may exist unit decompositions in $\Sigma(k)^\otimes$ that are not $\mathcal{R}(k)$ -equivalent but that are $\mathcal{R}(k')$ -equivalent for some $k' > k$.

► **Lemma 9.** *Let $k \in \mathbb{N}$ and \mathbf{U} and \mathbf{U}' be unit decompositions in $\Sigma(k)^\otimes$. If \mathbf{U} is $\mathcal{R}(k)$ -equivalent to \mathbf{U}' , then $\mathring{\mathbf{U}}$ is isomorphic to $\mathring{\mathbf{U}'}$.*

Proof. It is enough to show that if \mathbf{U} can be transformed into \mathbf{U}' in one $\mathcal{R}(k)$ -rewriting step then $\mathring{\mathbf{U}}$ is isomorphic to $\mathring{\mathbf{U}'}$. Therefore, assume that $\mathbf{U} \rightarrow \mathbf{U}'$. Then there exist unit decompositions $\mathbf{W}, \mathbf{W}' \in \Sigma(k)^\otimes$ and a rewriting rule $\mathbf{S}_1\mathbf{S}_2 \rightarrow \mathbf{S}'_1\mathbf{S}'_2$ in $\mathcal{R}(k)$ such that $\mathbf{U} = \mathbf{W}\mathbf{S}_1\mathbf{S}_2\mathbf{W}'$ and $\mathbf{U}' = \mathbf{W}\mathbf{S}'_1\mathbf{S}'_2\mathbf{W}'$. Since $\mathbf{S}_1 \circ \mathbf{S}_2 \sim \mathbf{S}'_1 \circ \mathbf{S}'_2$, we have an isomorphism φ from $\mathbf{S}_1 \circ \mathbf{S}_2$ to $\mathbf{S}'_1 \circ \mathbf{S}'_2$ that acts as the identity map on frontier vertices. This implies that $\mathring{\mathbf{U}} = \mathring{\mathbf{W}} \circ \mathbf{S}_1 \circ \mathbf{S}_2 \circ \mathring{\mathbf{W}'}$ is isomorphic to $\mathring{\mathbf{U}'} = \mathring{\mathbf{W}} \circ \mathbf{S}'_1 \circ \mathbf{S}'_2 \circ \mathring{\mathbf{W}'}$. ◀

Twisting. Let $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2 \cdots \mathbf{S}_n$ and $\mathbf{U}' = \mathbf{S}'_1\mathbf{S}'_2 \cdots \mathbf{S}'_n$ be two unit decompositions. We say that \mathbf{U} is a *twisting* of \mathbf{U}' if $\mathring{\mathbf{U}} = \mathring{\mathbf{U}'}$. Note that we are not equating slices up to isomorphism. In other words, we are really requiring that the slices $\mathring{\mathbf{U}}$ and $\mathring{\mathbf{U}'}$ are syntactically identical.

Let \mathbf{S}_1 and \mathbf{S}_2 be unit slices in $\Sigma(k)$ such that the out-frontier of \mathbf{S}_1 and the in-frontier of \mathbf{S}_2 have k' vertices for some $k' \leq k$. Given a permutation $\pi : [k'] \rightarrow [k']$, let \mathbf{S}_1^π be the slice obtained by renaming each vertex $(1, i)$ in the out-frontier of \mathbf{S}_1 to $(1, \pi(i))$, and let ${}^\pi\mathbf{S}_2$

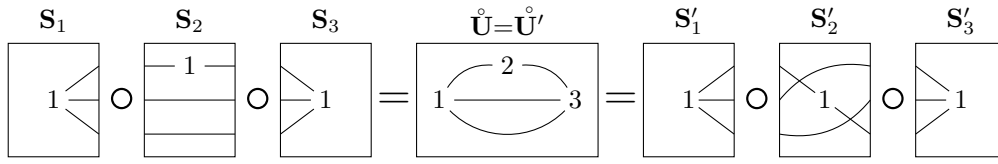


Figure 8 The unit decomposition $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2\mathbf{S}_3$ is a twisting of the unit decomposition $\mathbf{U}' = \mathbf{S}'_1\mathbf{S}'_2\mathbf{S}'_3$. Note that $\mathbf{S}_2 \circ \mathbf{S}_3 = \mathbf{S}'_2 \circ \mathbf{S}'_3$. Note that if we let π be the permutation that sets $\pi(1) = 2$, $\pi(2) = 3$ and $\pi(3) = 1$, then \mathbf{S}'_2 is obtained by permuting the out-frontier of \mathbf{S}_2 according to π and \mathbf{S}'_3 is obtained by permuting the in-frontier of \mathbf{S}_3 according to π .

be the slice obtained by renaming each vertex $(0, i)$ in the in-frontier of \mathbf{S}_2 to $(0, \pi(i))$. Then it should be clear that $\mathbf{S}_1 \circ \mathbf{S}_2 = \mathbf{S}'_1 \circ \pi\mathbf{S}_2$. Additionally, for each two unit slices \mathbf{S}'_1 and \mathbf{S}'_2 such that $\mathbf{S}_1 \circ \mathbf{S}_2 = \mathbf{S}'_1 \circ \mathbf{S}'_2$, it should be clear that there is some permutation π such that $\mathbf{S}'_1 = \mathbf{S}'_1 \circ \pi$ and $\mathbf{S}'_2 = \pi\mathbf{S}_2$. Note also that for each two such slices \mathbf{S}'_1 and \mathbf{S}'_2 , the rewriting rule $\mathbf{S}_1\mathbf{S}_2 \rightarrow \mathbf{S}'_1\mathbf{S}'_2$ belongs to $\mathcal{R}(k)$. This implies that if a unit decomposition $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2 \dots \mathbf{S}_n$ is a twisting of a unit decomposition $\mathbf{U}' = \mathbf{S}'_1\mathbf{S}'_2 \dots \mathbf{S}'_n$, then \mathbf{U} and \mathbf{U}' are $\mathcal{R}(k)$ -equivalent and can be transformed into each other by applying a sequence of rewriting rules that “twists” for each $i \in [n - 1]$ the out-frontier of \mathbf{S}_i and the in-frontier of \mathbf{S}_{i+1} according to some permutation π_i . This process is illustrated in Figure 8.

► **Proposition 10** (Twisting). *Let $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2 \dots \mathbf{S}_n$ and $\mathbf{U}' = \mathbf{S}'_1\mathbf{S}'_2 \dots \mathbf{S}'_n$ be two unit decompositions in $\Sigma(k)^\otimes$ such that \mathbf{U} is a twisting of \mathbf{U}' . Then, \mathbf{U} can be transformed into \mathbf{U}' by the application of $n - 1$ rewriting rules from $\mathcal{R}(k)$.*

An interesting question is whether, for each $k \in \mathbb{N}$, there is some $k' \in \mathbb{N}$ such that any two unit decompositions \mathbf{U} and \mathbf{U}' in $\Sigma(k)$ are $\mathcal{R}(k')$ -equivalent if and only if $\hat{\mathbf{U}}$ is isomorphic to $\hat{\mathbf{U}}'$. The answer turns out to be yes, as shown in Theorem 11 below.

► **Theorem 11.** *Let \mathbf{U} and \mathbf{U}' be unit decompositions in $\Sigma(k)^\otimes$. Then, $\hat{\mathbf{U}}$ is isomorphic to $\hat{\mathbf{U}}'$ if and only if \mathbf{U} and \mathbf{U}' are $\mathcal{R}(2k)$ -equivalent.*

Proof. Let $\mathbf{U} = \mathbf{S}_1\mathbf{S}_2 \dots \mathbf{S}_n$ and $\mathbf{U}' = \mathbf{S}'_1\mathbf{S}'_2 \dots \mathbf{S}'_n$. Suppose that \mathbf{U} and \mathbf{U}' are $\mathcal{R}(2k)$ -equivalent. Then by Lemma 9, $\hat{\mathbf{U}}$ is isomorphic to $\hat{\mathbf{U}}'$.

For the converse, suppose that $\hat{\mathbf{U}}$ is isomorphic to $\hat{\mathbf{U}}'$ and let φ be an isomorphism from $\hat{\mathbf{U}}$ to $\hat{\mathbf{U}}'$. We show that \mathbf{U} and \mathbf{U}' are $\mathcal{R}(2k)$ -equivalent.

Given a position $i \in [n - 1]$ in the unit decomposition \mathbf{U} , a *swap* between \mathbf{S}_i and \mathbf{S}_{i+1} is a rewriting rule in $\mathcal{R}(k')$ for some k' that rewrites \mathbf{U} into the unit decomposition

$$\mathbf{U}_i = \mathbf{S}_1\mathbf{S}_2 \dots \mathbf{S}_{i-1}\mathbf{S}''_i\mathbf{S}''_{i+1}\mathbf{S}_{i+2} \dots \mathbf{S}_n$$

such that, the function $\psi : [n] \rightarrow [n]$ that sets $\psi(p) = p$ for all $p \notin \{i, i + 1\}$, $\psi(i) = i + 1$ and $\psi(i + 1) = i$ is an isomorphism from $\hat{\mathbf{U}}$ to $\hat{\mathbf{U}}_i$.

Intuitively, we swap the center vertex of \mathbf{S}_i with the center vertex of \mathbf{S}_{i+1} . Note that there may be several rewriting rules corresponding to such a swap. Now, a swap in the unit decomposition \mathbf{U} corresponds to a swap in $\omega_{\mathbf{U}}$ as defined for linear orders in Section 2. The isomorphism φ defines a transformation of $\omega_{\mathbf{U}}$ into $\omega_{\mathbf{U}'}$.

By Proposition 7, $\text{cw}(\hat{\mathbf{U}}, \omega_{\mathbf{U}}) \leq k$ and $\text{cw}(\hat{\mathbf{U}}', \omega_{\mathbf{U}'}) \leq k$. Now, our result in Section 3 can be used for the slice rewriting system $\mathcal{R}(2k)$. More precisely, it follows from Theorem 3 that we can transform $\omega_{\mathbf{U}}$ into $\omega_{\mathbf{U}'}$ by a sequence of $O(n^2)$ swaps and at each step, the cutwidth is at most $2k$. By using the rewriting rules from $\mathcal{R}(2k)$, we can replicate these swaps into the unit decomposition \mathbf{U} , obtaining in this way a unit decomposition \mathbf{U}'' such that $\omega_{\mathbf{U}''} = \omega_{\mathbf{U}'}$.

8:12 Order Reconfiguration Under Width Constraints

Since $\mathring{U}'' = \mathring{U}'$, we have that U'' is a twisting of U' . Therefore, it follows from Proposition 10 that U'' can be further transformed into U' by applying a sequence of $n - 1$ rewriting rules from $\mathcal{R}(k) \subseteq \mathcal{R}(2k)$.

Hence, U can be rewritten into U'' by applying $O(n^2)$ rewriting rules from $\mathcal{R}(2k)$. ◀

Theorem 11 allows us to establish connections between the graph isomorphism problem for graphs of cutwidth at most k and the reachability problem in $\mathcal{R}(2k)$.

► **Theorem 12** ([10]). *Let G be an n -vertex graph of cutwidth k . We can compute a linear order ω of the vertices of G of width k in time $2^{\mathcal{O}(k^2)} \cdot n$.*

► **Theorem 13.** *Graph isomorphism for n -vertex graphs of cutwidth at most k can be reduced in time $2^{\mathcal{O}(k^2)} \cdot n$ to $\mathcal{R}(2k)$ -reachability.*

Proof. Given n -vertex graphs G and G' of cutwidth at most k , we first compute in time $2^{\mathcal{O}(k^2)} \cdot n$ linear orders ω and ω' of the vertex sets of G and G' , respectively, of cutwidth at most k . Then, from Proposition 8, we construct unit decompositions U and U' such that $\omega_U = \omega$, $\omega_{U'} = \omega'$, G is isomorphic to \mathring{U} and G' is isomorphic to \mathring{U}' . By Proposition 8, those decompositions belong to $\Sigma(k)^{\otimes}$. By Theorem 11, we have that \mathring{U} and \mathring{U}' are isomorphic if and only if U and U' are $\mathcal{R}(2k)$ -equivalent. ◀

5 Order Reconfiguration Parameterized by Vertex Separation Number

In this section, we show that the techniques employed in Section 3 for total orders of bounded cutwidth can be generalized to the context of orders of bounded vertex-separation number (Theorem 16). We consider that this generalization may be of independent interest in the theory of reconfiguration, since vertex separation number is a width measure for graphs that is strictly more expressive than cutwidth.

Let G be an n -vertex graph. Given sets $S, S' \subseteq V(G)$, we let $V(G, S, S') = \{u \in S : \exists v \in S' : \{u, v\} \in E(G)\}$ be the set of vertices in S that are adjacent to some vertex in S' . As a special case, we define $V(G, S) = V(G, S, V(G) \setminus S)$. We will often make use of the following monotonicity property without explicitly mentioning: If $T \subseteq S$ and $T' \subseteq S'$, then $|V(G, T, T')| \leq |V(G, S, S')|$.

► **Definition 14** (Vertex Separation Number). *Let G be an n -vertex undirected graph with vertex set $V(G)$ and edge set $E(G)$. Let $\omega : [n] \rightarrow V(G)$ be a linear order on the vertices of G . For each $p \in [n]$, we let*

$$\text{vsn}(G, \omega, p) = |V(G, \omega([p-1]))| = |\{l \in [p-1] : \exists r \geq p \text{ such that } \{\omega(l), \omega(r)\} \in E(G)\}|.$$

The vertex separation number of ω is defined as $\text{vsn}(G, \omega) = \max_{p \in [n]} \text{vsn}(G, \omega, p)$. The vertex separation number of G is defined as $\text{vsn}(G) = \min_{\omega} \text{vsn}(G, \omega)$, where ω ranges over all linear orders on the vertex set V .

For each $k \in \mathbb{N}$ and each n -vertex graph G , let $\text{VSN}(G, k) = \{\omega : [n] \rightarrow V(G) : \text{vsn}(G, \omega) \leq k\}$ be the set of linear orders of $V(G)$ of vertex separation number at most k . We say that ω can be *reconfigured* into ω' in vertex separation number at most k if there is a *reconfiguration sequence* $\omega = \omega_0 \xrightarrow{i_1} \omega_1 \xrightarrow{i_2} \dots \xrightarrow{i_r} \omega_r = \omega'$ such that for each $j \in [r]$, $\omega_j \in \text{VSN}(G, k)$.

► **Problem 15** (Bounded Vertex Separation Number Reconfiguration). *Let G be an n -vertex graph, $\omega, \omega' : [n] \rightarrow V(G)$ be linear orders on the vertex set of G , and $k \in \mathbb{N}$. Is it true that ω can be reconfigured into ω' in vertex separation number at most k ?*

The proof of Theorem 16 below is analogous to the proof of Theorem 3. More precisely, this proof follows by restating Lemma 4, Lemma 5 and Lemma 6 in terms of the vertex separation number of a graph instead of cutwidth, and then by using this last restated lemma to conclude the proof, as done in Theorem 3.

► **Theorem 16.** *Let G be an n -vertex graph and $\omega, \omega' : [n] \rightarrow V(G)$ be linear orders of $V(G)$ of vertex separation number at most k . Then, ω can be reconfigured into ω' in vertex separation number at most $\text{vsn}(G, \omega) + \text{vsn}(G, \omega') \leq 2k$.*

6 Conclusion

In this work, we have studied the order reconfiguration problem under the framework of the theory of fixed-parameter tractability. In particular, in our main technical result, we have shown that the order reconfiguration problem for orders of cutwidth at most k can always be achieved in cutwidth at most $2k$ (Theorem 3). Using this result, we have established new connections between the graph isomorphism problem and the reachability problem for a special two-letter string rewriting system operating on unit slices. In particular, we have proven that unit decompositions \mathbf{U} and \mathbf{U}' of width k are $R(2k)$ -equivalent if and only if the graphs \mathbf{U} and \mathbf{U}' are isomorphic (Theorem 11).

Theorem 11 opens up the possibility of studying the graph isomorphism problem under the perspective of term rewriting theory. The most immediate question in this direction is the complexity of deciding $R(2k)$ -reachability for unit decompositions of width k . By a reduction to isomorphism of graphs of cutwidth k , this problem can be solved in time $2^{O(k \cdot \text{polylog } k)} n^{O(1)}$ using the results from [14]. Can techniques that are intrinsic to string/term rewriting theory be used to improve this running time? Can such techniques be used to obtain algorithms running in time $f(k) \cdot n^{O(1)}$ for some computable function $f : \mathbb{N} \rightarrow \mathbb{N}$? Note that a positive answer to this question would be conceptually relevant even if the function $f(k)$ is substantially worse than the current $2^{O(k \cdot \text{polylog } k)}$, since techniques based on rewriting may carry over to contexts where group theoretic techniques do not. One interesting line of attack for this question would be to study connections between $R(2k)$ and techniques related to Knuth-Bendix completion and their generalizations [25, 26, 18, 16].

A natural question that arises in the context of reconfiguration of linear orders is the following: given two linear orders ω and ω' , what is the minimum cutwidth of a linear order ω'' occurring in a reconfiguration sequence from ω to ω' ? Is this problem NP-hard, or hard to approximate? Is it solvable in FPT-time for certain parameters? We thank one of the reviewers for bringing these interesting questions to our attention.

References

- 1 Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.
- 2 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 684–697. ACM, 2016.
- 3 László Babai, D. Yu. Grigoryev, and David M. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC '82*, pages 310–324. ACM, 1982.

- 4 László Babai, William M. Kantor, and Eugene M. Luks. Computational complexity and the classification of finite simple groups. In *24th Annual Symposium on Foundations of Computer Science, FOCS '83*, pages 162–171. IEEE, 1983.
- 5 Erik Barendsen. *Term Rewriting Systems*. Cambridge University Press, 2003.
- 6 Ronald V. Book and Friedrich Otto. String-rewriting systems. In *String-Rewriting Systems*, pages 35–64. Springer, 1993.
- 7 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.
- 8 John A. Ellis, Ivan H. Sudborough, and Jonathan S. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994.
- 9 Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. Polynomial-time algorithms for permutation groups. In *21st Annual Symposium on Foundations of Computer Science, FOCS '80*, pages 36–41. IEEE Computer Society, 1980.
- 10 Archontia C. Giannopoulou, Michał Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: obstructions and algorithmic aspects. *Algorithmica*, 81(2):557–588, 2019.
- 11 Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *Journal of the ACM*, 59(5):27, 2012.
- 12 Martin Grohe and Dániel Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM Journal on Computing*, 44(1):114–159, 2015.
- 13 Martin Grohe, Daniel Neuen, and Pascal Schweitzer. A faster isomorphism test for graphs of small degree. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 89–100. IEEE, 2018.
- 14 Martin Grohe, Daniel Neuen, Pascal Schweitzer, and Daniel Wiebking. An improved isomorphism test for bounded-tree-width graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, pages 67:1–67:14, 2018.
- 15 Martin Grohe and Pascal Schweitzer. Isomorphism testing for graphs of bounded rank width. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 1010–1029. IEEE, 2015.
- 16 Nao Hirokawa, Aart Middeldorp, Christian Sternagel, and Sarah Winkler. Abstract completion, formalized. *Logical Methods in Computer Science*, 15(3), 2019.
- 17 Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.
- 18 Deepak Kapur and Paliath Narendran. The Knuth-Bendix completion procedure and Thue systems. *SIAM Journal on Computing*, 14(4):1052–1072, 1985.
- 19 Jan Willem Klop. *Term Rewriting Systems*. Centrum voor Wiskunde en Informatica, 1990.
- 20 Stefan Kratsch and Pascal Schweitzer. Isomorphism for graphs of bounded feedback vertex set number. In Haim Kaplan, editor, *Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory*, volume 6139 of *Lecture Notes in Computer Science*, pages 81–92. Springer, 2010.
- 21 Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 186–195. IEEE, 2014.
- 22 Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.
- 23 Gary Miller. Isomorphism testing for graphs of bounded genus. In Raymond E. Miller, Seymour Ginsburg, Walter A. Burkhard, and Richard J. Lipton, editors, *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, STOC '80*, pages 225–235. ACM, 1980.
- 24 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.

- 25 Christian Sternagel and René Thiemann. Formalizing Knuth-Bendix orders and Knuth-Bendix completion. In Femke van Raamsdonk, editor, *24th International Conference on Rewriting Techniques and Applications, RTA 2013*, volume 21 of *LIPICs*. Schloss Dagstuhl, Leibniz-Zentrum für Informatik, 2013.
- 26 Ian Wehrman, Aaron Stump, and Edwin Westbrook. Slothrop: Knuth-Bendix completion with a modern termination checker. In Frank Pfenning, editor, *Term Rewriting and Applications, 17th International Conference, RTA 2006*, volume 4098 of *Lecture Notes in Computer Science*, pages 287–296. Springer, 2006.
- 27 Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *Journal of Computer and System Sciences*, 93:1–10, 2018.

Universal Gauge-Invariant Cellular Automata

Pablo Arrighi 

Université Paris-Saclay, CNRS, LMF, 91190 Gif-sur-Yvette, France
IXXI, Lyon, France

Marin Costes  

ENS Paris-Saclay, CNRS, LMF, 91190 Gif-sur-Yvette, France

Nathanaël Eon 

Aix-Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France

Abstract

Gauge symmetries play a fundamental role in Physics, as they provide a mathematical justification for the fundamental forces. Usually, one starts from a non-interactive theory which governs “matter”, and features a global symmetry. One then extends the theory so as to make the global symmetry into a local one (a.k.a. gauge-invariance). We formalise a discrete counterpart of this process, known as gauge extension, within the Computer Science framework of Cellular Automata (CA). We prove that the CA which admit a relative gauge extension are exactly the globally symmetric ones (a.k.a. the colour-blind). We prove that any CA admits a non-relative gauge extension. Both constructions yield universal gauge-invariant CA, but the latter allows for a first example where the gauge extension mediates interactions within the initial CA.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Cellular automata, Gauge-invariance, Universality

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.9

Funding This publication was made possible through the support of the ID# 61466 grant from the John Templeton Foundation, as part of the “The Quantum Information Structure of Spacetime (QISS)” Project (qiss.fr). The opinions expressed in this publication are those of the author(s) and do not necessarily reflect the views of the John Templeton Foundation.

Acknowledgements The authors would like to thank Guillaume Theyssier for asking us the question whether any CA admits a gauge extension.

1 Introduction

Symmetries are an essential concept, whether in Computer science or in Physics. In this paper we explore the Physics concept of gauge symmetry by taking it into the rigorous, Computer Science framework of Cellular Automata (CA). Implementing gauge-symmetries within CA may prove useful in the fields of numerical analysis; quantum simulation; and digital Physics – as these are constantly looking for discrete schemes that simulate known Physics. Quite often, these discrete schemes seek to retain the symmetries of the simulated Physics; whether in order to justify the discrete scheme as legitimate or as numerically accurate (e.g. by doing the Monte Carlo-counting right [12]). More specifically, the introduction of gauge-symmetries within discrete-time lattice models has proven useful already in the field of Quantum Computation, where gauge-invariant Quantum Walks and Quantum Cellular Automata [1] provide us with concrete digital quantum simulation algorithms for particle Physics. These come to complement the already existing continuous-time lattice models of particle Physics [9, 15]. Another field where this has played a role is Quantum error correction [13, 14], where it was noticed that gauge-invariance amounts to invariance under certain local errors. This echoes the fascinating albeit unresolved question of noise resistance within Cellular Automata [7, 8, 11, 18].



© Pablo Arrighi, Marin Costes, and Nathanaël Eon;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 9; pp. 9:1–9:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In [16] the authors study G -blind cellular automata, where G is a group of permutation acting on the state space of cells. Blind cellular automata are globally symmetric under G , i.e. the global evolution commutes with the application of the same $g \in G$ at once on every cell. They show the surprising result that any CA can be simulated by such a globally symmetric CA, when G is the symbol permutations. Globally symmetric CA are therefore universal. Local symmetry, aka gauge symmetry, is way more stringent however: a different g^x can now be chosen for every cell x . Still, in this paper, we prove that any CA can be extended into a gauge-invariant CA. Gauge-invariant CA are therefore universal.

From a Physics perspective one usually motivates the demand for a certain gauge symmetry, from an already existing global symmetry. From a mathematical perspective, the gauge field that then gets introduced for that purpose is often seen as a connection between two gauge choices at neighbouring points. This raises questions however, because there is no immediate reason why a gauge symmetry should necessarily arise from an already existing global symmetry (one could ask for a certain ad hoc gauge symmetry from scratch). Nor is there an immediate reason why a gauge field should necessarily be interpretable as a connection (a gauge field could be made to hold absolute instead of relative information about gauge choices).

In this paper, we prove an original result relating these two folklore perspectives about gauge theories using purely combinatorial definitions. Namely, we prove that the CA that admit *relative* gauge extension are exactly those that have the corresponding global symmetry in the first place.

Although the gauge field was initially introduced in order to obtain gauge symmetry, it allows for new dynamics. Amongst those dynamics, one could ask for the matter field to influence the dynamics of the gauge field, as is the case in Physics. In this paper, we provide a first a Gauge-invariant CA where this is happening. This CA is obtained through a non-relative gauge extension. We leave it as an open question whether the same can be achieved through a relative gauge extension.

The present work builds upon two previous papers by a subset of the authors, which laid down the basic definitions of gauge-invariance for CA and provided a first set of examples, in both the abelian [2] and the non-abelian [3] cases. Sec. 2 first recalls these basic definitions, but it also formalises the notions of general and relative gauge extensions, which were still missing. Sec. 3 shows that CA admit a relative gauge extension if and only if they are globally symmetric. Sec. 4 shows that any CA admits a general gauge extension. Sec. 5 draws the consequences upon universality. Sec. 6 provides a first example of a gauge-extended CA whose gauge field is sourced by the matter field.

2 Definitions

2.1 Cellular Automata

A cellular automaton (CA) consist in an array of identical cells, each of which may take one in a finite number of possible states. The whole array evolves in discrete time steps by iterating a function F . Moreover this global evolution F is shift-invariant (it acts everywhere the same) and causal (information cannot be transmitted faster than some fixed number of cells per time step). Let us make this formal.

► **Definition 2.1** (Configuration). *A configuration c over an alphabet Σ and a space \mathbb{Z}^d is a function that associates a state to each point:*

$$c : \mathbb{Z}^d \longrightarrow \Sigma$$

The set of all configurations will be denoted $\Sigma^{\mathbb{Z}^d}$.

A configuration should be seen as the state of the CA at a given time. We use the short-hand notation $c_x = c(x)$ for $x \in \mathbb{Z}^d$ and $c|_I$ for the configuration c restricted to the set I , i.e. $c : I \rightarrow \Sigma$, for $I \subseteq \mathbb{Z}^d$. The association of a position and its state is called a cell.

The way to describe a global evolution F that is causal, is via the provision of a local rule. A local rule takes as input a configuration restricted to $x + \mathcal{N}$, and outputs the next value of the cell x , i.e. $f : \Sigma^{\mathcal{N}} \rightarrow \Sigma$, where \mathcal{N} is a finite subset of \mathbb{Z}^d referred to as “the neighbourhood”. Applying f at every position x simultaneously, implements F .

► **Definition 2.2** (Cellular automata). *The CA F having neighbourhood \mathcal{N} and local function $f : \Sigma^{\mathcal{N}} \rightarrow \Sigma$ is the function $F : \Sigma^{\mathbb{Z}^d} \rightarrow \Sigma^{\mathbb{Z}^d}$ such that for all $x \in \mathbb{Z}^d$,*

$$F(c)_x = f(c|_{x+\mathcal{N}}).$$

We sometimes denote by $c_{t,x}$ the value of a cell at position x and time t , where $c_{t+1} = F(c_t)$.

2.2 Global versus gauge symmetry

Global symmetry

We say that a CA is globally symmetric whenever its global evolution is invariant under the application of the same alphabet permutation at every position at once. Globally symmetric CA are also known as G -blind CA [16] with G a group of permutations over Σ .

► **Definition 2.3** (Globally symmetric). *Let $F : \Sigma^{\mathbb{Z}^n} \rightarrow \Sigma^{\mathbb{Z}^n}$ be a CA and G be a group of permutations over Σ . For all $g \in G$, let \bar{g} denote its application at every position simultaneously: $\bar{g}(c)_i = g(c_i)$. We say that F is globally G -symmetric if and only if, for any g in G , we have $F \circ \bar{g} = \bar{g} \circ F$.*

Local/gauge symmetry

We say that a CA is locally symmetric whenever its global evolution is invariant under the application of a local permutation at every position. The first difference with globally symmetric CA is the permutation is now allowed to differ from one position to the next. The second difference is that the permutation is now allowed to act on the surrounding cells. Locally symmetric CA are referred to as gauge-invariant CA [2, 3, 4].

► **Definition 2.4** (Local gauge-transformation group). *Let g be a permutation over $\Sigma^{(2s+1)^d}$, with $s \in \mathbb{N}$. We denote by $g^x : \Sigma^{\mathbb{Z}^d} \rightarrow \Sigma^{\mathbb{Z}^d}$ the function that acts as g on the cells at $\llbracket x-s, x+s \rrbracket^d$, and trivially everywhere else. A local gauge-transformation group G is a group of bijections over $\Sigma^{(2s+1)^d}$, such that for any $g, h \in G$ and any $x \neq y \in \mathbb{Z}^d$, $g^x \circ h^y = h^y \circ g^x$.*

This permutation condition makes it irrelevant to consider which local gauge-transformation gets applied first, so that the product $g^x h^y$ be defined. The condition is decidable, checking it over the $\llbracket -2s, +2s \rrbracket^d$ suffices.

► **Definition 2.5** (Gauge-transformation). *Consider G a group of local gauge-transformations. A gauge-transformation is then specified by a function $\gamma : \mathbb{Z}^d \rightarrow G$. It is interpreted as acting over $c \in \Sigma^{\mathbb{Z}^d}$ as follows:*

$$\gamma(c) = \left(\prod_{x \in \mathbb{Z}^d} \gamma^x \right)(c),$$

where γ^x is short for $\gamma(x)^x$. We denote by Γ the set of gauge-transformations.

Notice how an element $\gamma \in \Gamma$ may be thought of as a configuration over the alphabet G . Thus γ_x is an element of G which can be applied on a finite configuration, while γ^x is its natural extension which can be applied onto a full configuration.

Gauge-invariant CA are “insensitive” to gauge-transformations: performing γ before F amounts to performing some γ' after F .

► **Definition 2.6** (Gauge-invariant CA). *Let F be a CA, G be a local gauge-transformation group, and Γ be the corresponding set of gauge-transformations. F is Γ -gauge-invariant if and only if there exists a CA Z over the alphabet G , such that for all $\gamma \in \Gamma$:*

$$Z(\gamma) \circ F = F \circ \gamma.$$

The reason why γ' must result from a CA Z , instead of being left fully arbitrary, is because F is deterministic, shift-invariant and causal – from which it follows that γ' , if it exists, can be computed deterministically, homogeneously and causally from the γ applied before. Thus, the above is demanding a weakened commutation relation between the evolution F and the set of gauge-transformations Γ . In practice in Physics Z is often the identity, making gauge-invariance a commutation relation. This will be the case in our constructions.

2.3 (Relative) gauge extensions

In Physics, one usually begins with a theory that explains how matter freely propagates, i.e. in the absence of forces. This initial theory solely concerns the “matter field”, and is not gauge-invariant. For instance, the Dirac equation, which dictates how electrons propagate, is not $U(1)$ -gauge-invariant. Next, one enriches the initial theory with a second field, the so-called gauge field, so as to make the resulting theory gauge-invariant. For instance, the case of the electron, $U(1)$ -gauge-invariance is obtained thanks to the addition of the electromagnetic field. The resulting theory can still account for the free propagation of the matter field, but the presence of the gauge field also allows for richer behaviours, e.g. electromagnetism. Quite surprisingly three out of the four fundamental forces can be introduced mathematically, and thereby justified by gauge symmetry requirements, through this process of “gauge extension”.

But when is it the case that a theory is a gauge extension of another, exactly? In Physics this is left informal. One of the contributions of this paper is the provide a first rigorous definition of the notion of gauge extension, and of its relative subcase, in the discrete context of CA.

General gauge extension

A gauge extension must simulate the initial CA, extend the required gauge-transformations, and achieve gauge-invariance overall:

► **Definition 2.7** (Gauge extension). *Let F be a CA over alphabet Σ . Let Γ be a gauge-transformation group over Σ . Let Λ be a finite set which will serve as the gauge field alphabet. A gauge extension of (F, Γ) is a tuple (F', Γ') with F' a CA over alphabet $\Sigma \times \Lambda$ and Γ' a gauge-transformation group over $\Sigma \times \Lambda$, such that:*

- (Simulation) *there exists $\epsilon \in \Lambda$ such that F' simulates one step of F when the gauge field value is set ϵ everywhere. In other words for any $c \in \Sigma^{\mathbb{Z}^d}$, there exists $e' \in \Lambda^{\mathbb{Z}^d}$,*

$$F'(c, e) = (F(c), e')$$

where e is the constant gauge field configuration ($x \mapsto \epsilon$).

- (Extension) Γ' extends Γ : there exists a bijection $B : \Gamma' \rightarrow \Gamma$ such that for any $\gamma' \in \Gamma'$, there exists a CA $L_{\gamma'}$ over alphabet Λ , such that for any $c, \lambda \in \Sigma^{\mathbb{Z}^d} \times \Lambda^{\mathbb{Z}^d}$,

$$\gamma'(c, \lambda) = (\gamma(c), L_{\gamma'}(\lambda)) \tag{1}$$

where $\gamma = B(\gamma')$.

- (Gauge-invariance) F' is Γ' -gauge-invariant.

We used implicitly the canonical bijection between $(\Sigma \times \Lambda)^{\mathbb{Z}^d}$ and $\Sigma^{\mathbb{Z}^d} \times \Lambda^{\mathbb{Z}^d}$

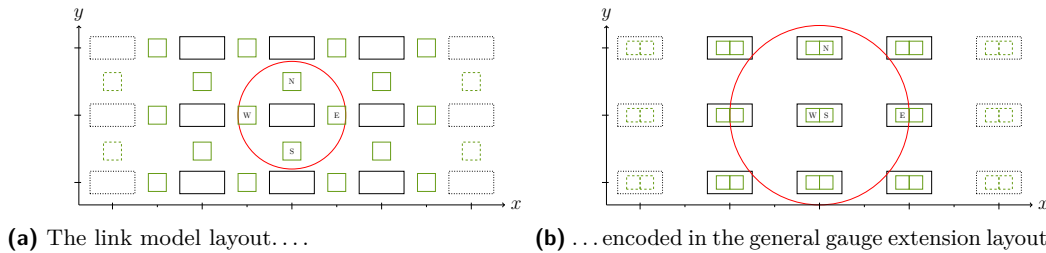
Notice that when the gauge field does not evolve in time, we can rewrite the simulation condition as $F'(c, e) = (F(c), e)$. Then F is a sub-automaton of F' [10], whenever the gauge field is set to e .

Intuitively, the gauge field’s role is to keep track of which gauge-transformation got applied where, so as to hold enough information to insure gauge-invariance. There are different ways to do this; for instance one could indeed store the “gauge” at each point, i.e. which gauge-transformation has happened at the specific point. But one could be more parsimonious and store just the “relative gauge”, i.e. which gauge-transformation relates that which has happened at every two neighbouring points.

Relative gauge extension

The standard choice in the Physics literature is to place the gauge field between the matter cells only – i.e. on the links between two cells. This choice of layout is sometimes referred to as the “quantum link model” [6, 17]. The mathematical justification for this choice, is precisely that the gauge field may be interpreted as relative information between neighbouring matter cells. Geometrically speaking, it may be understood as a “connection” relating two closely “tangent spaces” on a manifold.

Our previous definition of general gauge extensions does allow for such relative gauge extensions as a particular case, up to a slight recoding, as shown in Fig-1, i.e. the link model is simulated by transferring the value of a gauge field on a link, to the vertex at the tip of the link.



■ **Figure 1** Capturing the link model used for relative gauge extensions with the general definition.

The following specialises the previous, mathematical notion of gauge extension, to the restricted way in which it is understood in Physics:

► **Definition 2.8** (Relative gauge extension). Given a CA F and a local gauge-transformation group Γ of radius $s = 0$, we say that a gauge extension (F', Γ') is relative when:

- (a) the gauge field is positioned on the links $(x, x + e_d)$, where e_d takes values in $\{(1\ 0\ 0\ \dots), (0\ 1\ 0\ \dots), \dots\}$.
- (b) the gauge field takes values in G and the ϵ from Definition 2.7 is the identity – i.e. $\Lambda = G$ and $\epsilon = I$

(c) for every position x a gauge-transformations $\gamma \in \Gamma'$ act both on the matter field at x according to $\gamma_x \in G$, and on the gauge field a of its links, as follows:

$$\begin{cases} \gamma^x(a)_{(x-e_d,x)} &= \gamma_x \circ a_{(x-e_d,x)} \\ \gamma^x(a)_{(x,x+e_d)} &= a_{(x,x+e_d)} \circ \gamma_x^{-1} \end{cases} \quad (2)$$

Thus relative extension keeps track of the difference of gauge between two neighbouring cells.

The above definitions were given for the \mathbb{Z}^d grid, in order to establish the notion of gauge extension in full generality. The next section, however, will be given just in one dimension ($d = 1$) for clarity. We have established it in arbitrary dimension d in a private manuscript.

3 Globally symmetric CA admit a relative gauge extension

From a Physics perspective, the gauge symmetry one seeks to impose usually comes from an already existing global symmetry. We show here that there is an equivalence between being globally G -symmetric and having a gauge extension with respect to G a subgroup of the permutations of Σ .

► **Theorem 3.1** (Global symmetry and relative gauge extension). *Let F be a CA over alphabet Σ , G a subgroup of the permutations of Σ and Γ the set of gauge-transformations defined using G as the group of local gauge-transformations. Then the following two properties are equivalent:*

- (i) F is globally G -symmetric
- (ii) (F, Γ) admits a relative gauge extension (F', Γ') with the identity for the gauge field evolution, such that F' commutes with any element of Γ' (stronger than gauge-invariance because it does not require a Z -map).

Proof.

(i \Rightarrow ii). Let f be the local rule of F with radius r . Let F' be a CA of radius r over the extended configurations – containing a gauge field in between neighboring cells – such that the gauge field evolution is the identity and the local rule f' for the evolution of the matter field is defined as follows:

$$\begin{aligned} f'(c_{-r}, a_{(-r+1, -r)}, \dots, c_0, a_{(0,1)}, \dots, c_r) = \\ f \left(\prod_{i=-1}^{-r} a_{(i, i+1)}(c_{-r}), \dots, a_{(-1,0)}(c_{-1}), c_0, a_{(0,1)}^{-1}(c_1), \dots, \prod_{i=1}^r a_{(i-1, i)}^{-1}(c_r) \right) \end{aligned} \quad (3)$$

where a is the gauge field and

$$\begin{aligned} \prod_{i=-1}^{-j} a_{(i, i+1)}(c_{-j}) &= a_{(-1,0)} \circ \dots \circ a_{(-j+1, -j+2)} \circ a_{(-j, -j+1)}(c_{-j}) \\ \prod_{i=1}^j a_{(i-1, i)}^{-1}(c_j) &= a_{(0,1)}^{-1} \circ \dots \circ a_{(j-2, j-1)}^{-1} \circ a_{(j-1, j)}^{-1}(c_j). \end{aligned}$$

f' is defined so as to apply f on a configuration that is on the same gauge basis – i.e. where relative gauge-transformations are cancelled and every cell is looked through the eyes of the gauge at position 0.

We shall now prove that (F', Γ') – with Γ' defined through Eq.(2) – is a relative gauge extension of (F, Γ) .

The fact that this extension is relative is immediate from the definition. We therefore need to prove that this extension has the 3 required properties from definition 2.7.

- *(Simulation)* The fact that F' simulates F when the gauge field is the identity is immediate from the definition of f' .
- *(Extension)* Because Γ' is defined through definition 2.8, it is also immediate that it verifies the extension property.
- *(Gauge-invariance)* For any $\gamma' \in \Gamma'$ we will check that $\gamma' \circ F' = F' \circ \gamma'$. For j between 1 and r , we apply the gauge-transformation on the inputs (c and a) – using Eq.(2) – and obtain by simple computation the following, where $\gamma = B(\gamma')$ through definition 2.7:

$$\begin{cases} \prod_{i=1}^j [\gamma_i \circ a_{(i-1,i)} \circ \gamma_{i-1}^{-1}]^{-1} \gamma_j(c_j) & = \gamma_0 \circ \left[\prod_{i=1}^j a_{(i-1,i)}^{-1}(c_j) \right] \\ \prod_{i=-1}^{-j} [\gamma_{i+1} \circ a_{(i,i+1)} \circ \gamma_i^{-1}] \gamma_j(c_j) & = \gamma_0 \circ \left[\prod_{i=-1}^{-j} a_{(i,i+1)}(c_j) \right] \end{cases}$$

where $\gamma_i \in G$. Therefore using this and Eq.(3) we have that

$$(F' \circ \gamma'(c, a))_0 = \left(f \circ \gamma_0 \begin{pmatrix} \prod_{i=-1}^{-r} a_{(i,i+1)}(c_{-r}) \\ \dots \\ a_{(-1,0)}(c_{-1}) \\ c_0 \\ a_{(0,1)}^{-1}(c_1) \\ \dots \\ \prod_{i=r}^1 a_{(i-1,i)}^{-1}(c_r) \end{pmatrix} \right)_0 \quad (4)$$

where γ_0 is here applied to every element of the tuple. Since F is globally G -symmetric, we have that $f \circ \gamma_0 = \gamma_0 \circ f$ and therefore $(F' \circ \gamma'(c, a))_0 = \gamma_0 \circ (F'(c, a))_0$ which finishes the proof that $F' \circ \gamma' = \gamma' \circ F'$ through translation invariance and because the gauge field evolution is the identity.

(ii \Rightarrow i). Suppose that (F', Γ') is a relative gauge extension of (F, Γ) , such that F' commutes with any element of Γ' , we shall prove that F is globally G -symmetric (with Γ the gauge-transformation group based on G). Let c be a configuration and e denote the empty configuration of the gauge field. For any local gauge-transformation g , we write \bar{g} the global gauge-transformation applying g everywhere – g denotes both the element of G and G' depending on the context:

$$\begin{aligned} \bar{g} \circ F'(c, e) &= \bar{g}(F(c), a) && \text{(Simulation 2.7)} \\ &= (\bar{g}(F(c)), a') && \text{(Extension 2.7)} \end{aligned}$$

where a and a' can be any gauge field configuration depending on F' and \bar{g} . And

$$\begin{aligned} F' \circ \bar{g}(c, e) &= F'(\bar{g}(c), \bar{g} \circ e \circ \bar{g}^{-1}) && \text{(Extension 2.8)} \\ &= F'(\bar{g}(c), e) \\ &= (F(\bar{g}(c)), a') && \text{(Simulation 2.7)} \end{aligned}$$

where a and a' can be any gauge field configuration depending on F' and g . The G' -gauge-invariance of F' give us $\bar{g} \circ F'(c, e) = F' \circ \bar{g}(c, e)$ and thus

$$\bar{g}(F(c)) = F(\bar{g}(c)).$$

Therefore F is globally G -symmetric. ◀

This theorem proves useful when looking for relative gauge extensions: first search for a global symmetry. The construction will be used in Sec. 5 to prove that relative gauge extensions of CA are universal.

4 Non-globally symmetric CA still admit an absolute gauge extension

We now prove that any CA can be intrinsically simulated by a gauge-invariant one, with respect to any gauge-transformation group, of any radius. The construction of this section uses non-relative gauge extensions, but it allows us to get rid of the prior requirements that there be a global symmetry or that the gauge-transformations be of radius 0.

► **Theorem 4.1** (Every CA admits a gauge extension). *For any CA F and gauge-transformation group Γ there exists for some gauge field alphabet a gauge extension (F', Γ') . Furthermore the local rule of F' acts as the identity over the gauge field.*

Proof. We give here a constructive proof for any CA over \mathbb{Z}^d .

Let F be a CA of radius s' and G be a local gauge-transformation group of radius s . We denote r the highest radius between s and s' . In the following we will consider neighbourhoods $R_x^k = [x - k \cdot r, x + k \cdot r]^d$ of each point $x \in \mathbb{Z}^d$, with $[a, b] = \{n \in \mathbb{Z} \mid a \leq n \leq b\}$.

First we choose G as gauge field alphabet and define the effect of a global gauge-transformation γ^x as $\gamma^x(a)_x = \gamma_x \circ a_x$, where a denotes a gauge field configuration. The definition is so that the gauge field simply keeps track of every gauge-transformation applied around x . For any other cell of the gauge field, γ^x has no impact. This condition along with the extension property of Definition 2.7 fully defines the new gauge-transformation group Γ' .

Next we define a new local rule f' over the neighbourhood R_x^5 . The definition below just states that the local rule applies $\prod_{i \in R_x^2} a_i^{-1}$ to undo all previous gauge-transformations, it then computes the evolution of f , and finally reapplies all the gauge-transformations, i.e.

$$f'(c|_{R_x^5}, a|_{R_x^5}) = \left(\prod_{i \in R_x^1} a_i \circ f|_{R_x^2} \left(\prod_{i \in R_x^4} a_i^{-1} (c|_{R_x^5})|_{R_x^3} \right), a_x \right)$$

where $f|_{R_x^2}$ denotes the function from R_x^3 to R_x^2 which calculates the temporal evolution of our automaton.

We can rewrite this local rule globally, using the notation a to denote either the gauge field or a gauge-transformation which applies a_x around each position x :

$$F'(c, a)_x = \left(a \circ F \circ a^{-1}(c), a \right)_x$$

Let us check that (F', Γ') is a gauge extension:

- *(Simulation)* When the gauge field is the identity f' acts the same as f over the matter field, and as the identity over the gauge field.
- *(Extension)* We used this property to define G' .
- *(Gauge-invariance)* For any $\gamma' \in \Gamma'$ – where Γ' is built from G' through definition 2.5 – we must check that $\gamma' \circ F' = F' \circ \gamma'$. We reason globally to simplify notations:

$$F' \circ \gamma'(c, a) = F'(\gamma(c), \gamma(a)) \quad (\text{Extension 2.7})$$

$$= \left(\gamma(a) \circ F \circ \gamma(a)^{-1}(\gamma(c)), \gamma(a) \right) \quad (\text{Definition of } F')$$

$$= \left(\gamma \circ a \circ F \circ a^{-1} \circ \gamma^{-1} \circ \gamma(c), \gamma(a) \right) \quad (\text{Definition 2.4})$$

$$= \left(\gamma \circ a \circ F \circ a^{-1}(c), \gamma(a) \right) \quad (\text{Definition 2.4})$$

$$\gamma' \circ F'(c, a) = \gamma' \left(a \circ F \circ a^{-1}(c), a \right) \quad (\text{Definition of } F')$$

$$= \left(\gamma \circ a \circ F \circ a^{-1}(c), \gamma(a) \right) \quad (\text{Extension 2.7})$$

◀

5 (Relative) gauge-invariant CA are universal

Results in this section are only given for dimension 1.

In [16], the authors prove that for any alphabet Σ containing 2 symbols or more, there exists an intrinsically universal globally G -symmetric cellular automaton on $\Sigma^{\mathbb{Z}}$, where G is the group of all permutations of σ . The proof involves an extension which encodes the information in the structure of the configuration rather than the states, the idea being that a global transformation will conserve the structure – thus the information. Combining this result and Th. 3.1, we can easily prove the following corollary:

► **Corollary 5.1** (Relative gauge-invariant cellular automata are universal). *For any alphabet Σ with $|\Sigma| \geq 2$, any gauge transformation group G and any cellular automaton F on $\Sigma^{\mathbb{Z}}$, there exists a G' -gauge-invariant CA F' which intrinsically simulates F , with G' the extended gauge-transformation based on G . Moreover, F' arises as the relative gauge extension of a CA.*

Proof. Let F'' be a globally G -symmetric CA on $\Sigma^{\mathbb{Z}}$ that intrinsically simulates F using [16, Theorem 1]. From Th. 3.1, (F'', G) admits a relative gauge extension (F', G') with the evolution of the gauge field being the identity. Thus F' intrinsically simulates F'' , from which it follows that F' is a G' -gauge-invariant CA which intrinsically simulates F . ◀

Such result is interesting on two accounts: (i) it shows that universality only requires relative gauge information and does not need any absolute information stored in the gauge field; (ii) it shows that relative gauge extensions, which are the ones usually appearing in Physics, are universal. Still, the universality of gauge-invariant CA is an even more direct corollary of Th. 4.1. With that construction we can just pick any universal CA F , any local transformation group G of any radius, and gauge-extend F into F' . F' acts trivially on the gauge field in this construction, it thus intrinsically simulates F and is therefore universal.

6 Sourcing the gauge field with the matter field

In both the construction of Th. 3.1 and Th. 4.1, the evolution rule of the gauge field is the identity, meaning that it does not evolve with time. It is often the case in Physics that a further twist is then introduced, so that the matter field now influences the gauge field. We wish to do the same and find a gauge-extended CA whose gauge field influences the matter field, and whose matter field backfires on the gauge field.

We use here the general definition of a gauge extension (Definition 2.7) to search a gauge extension F' of a non gauge-invariant CA F . Without loss of generality, $F' = (F'_1, F'_2)$, where F'_1 takes (c, a) as input and returns the matter field after one time-step, and F'_2 does the same for the gauge field. We impose that the gauge (resp. matter) field be sourced by the matter (resp. gauge) field, in the strongest possible manner, i.e. we ask for F'_2 (resp. F'_1) to be injective in its first (resp. second) parameter.

We begin by choosing the alphabet $\Sigma = \{0, 1, 2\}^2$ and the space \mathbb{Z} and we denote by c_i^l and c_i^r respectively the left and the right part of the cell. In the following definitions we consider that all the additions and all the subtractions are modulo 3.

We define the initial automaton by the local rule: $F(c)_i = (c_{i-1}^l - c_i^r, c_i^l + c_{i+1}^r)$, cf. Fig.2.

We consider a local group of gauge-transformation containing three elements, namely:

$$G = \{\sigma_0, \sigma_1, \sigma_2\}$$

where σ_i is the function of radius 0 that adds i to each part of the cell.

9:10 Universal Gauge-Invariant Cellular Automata

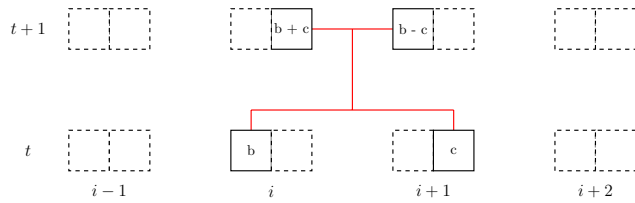


Figure 2 The local rule of F.

We can check that F is not gauge-invariant for Γ (as defined from G), by considering a configuration c which associates $(1, 1)$ to position i and $(0, 0)$ to all other positions. Let γ be a gauge-transformation which applies σ_2 over i , $\gamma(c)$ is then the fully empty configuration e . Since F preserves emptiness we have:

$$F \circ \gamma(c) = \gamma(c) = e$$

But when we apply F to c we obtain non-empty cells in $i + 1$ and $i - 1$, this contradicts the gauge-invariance definition. This idea is illustrated in sub-Figs 3a and 3b.

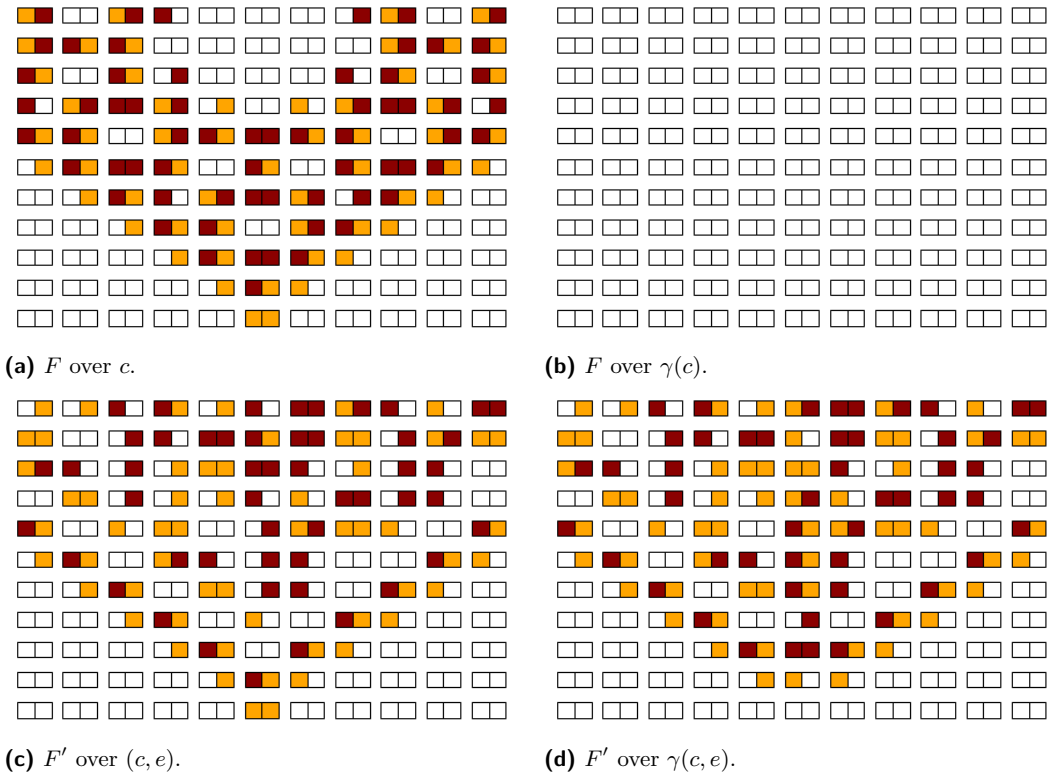


Figure 3 Space-time representation of F and F' over the same initial configurations c and $\gamma(c)$, where c is the configuration at the bottom line of 3a 3c. The values 1 and 2 are respectively represented by orange and red, while 0 is just an empty cell. Only the matter field is represented here.

We now provide a gauge extension for (F, Γ) . We begin by choosing the gauge field alphabet $\Lambda = \Sigma$ and placing the gauge field between each cell.

Next we engineer the injective influence of the gauge field over the matter field in the simplest possible way. We simply add, to each sub-cell of the matter field, the value of the nearest sub-cell of the gauge field during each evolution. See Fig. 4b.

Finally we extend gauge-transformations to the gauge field (Fig. 4a) and choose the evolution of the gauge field (green cells of Fig. 4b) to make sure that F' is a reversible gauge-invariant dynamics. Notice that the figures were chosen so that all cases are given. The proof of gauge-invariance for this example is given in appendix-A, and can be visually seen from Figs. 3c and 3d where a gauge-transformation does not impact the overall dynamics.

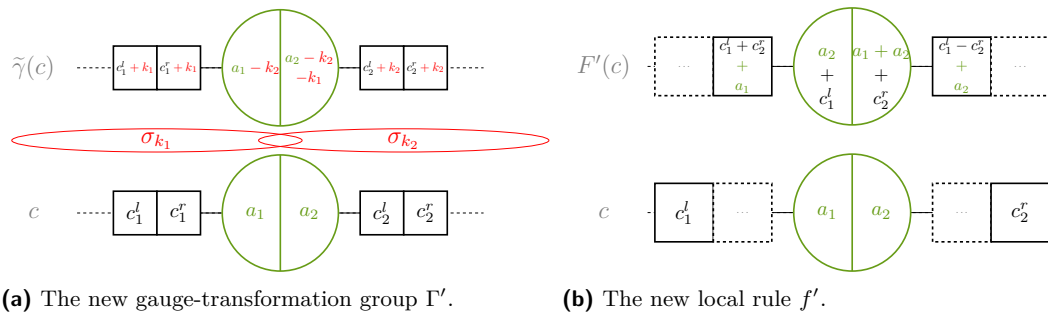


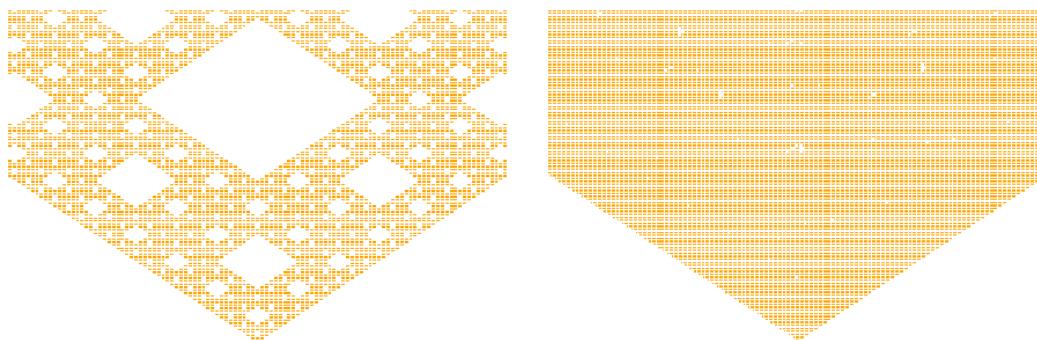
Figure 4 Description of the gauge extension (F', Γ') . Green circles represent the gauge field and black rectangles the matter field.

Overall, starting from a CA F we have defined a gauge extension F' which features a strong interaction between the gauge and the matter field. In the world of CA this is the first example of the kind [5, 3]. Building this example required the choice of a very specific extension of the gauge-transformation over the gauge field (cf Fig.4a) so as to obtain gauge-invariance whilst preserving reversibility and injectivity. Under a relative gauge extension this extension of the gauge-transformation is forced upon us, it seems hard to find such an example.

Notice that since the gauge field is sourced by the matter field it typically does not remain empty during the evolution. Thus F' can only simulate F for one time step. This may seem strange from a mathematical point of view, as we may expect from an extension that it preserves the original dynamics over several steps, too. But in Physics the initial non gauge-invariant theory is indeed used to inspire a more complex dynamics, which enriches and ultimately diverges from the original one. Fig.5 shows how starting from the same configuration, one obtains very different evolutions.

7 Conclusion

In order to obtain a gauge-invariant theory, starting from a non-gauge-invariant one, the usual route is to extend the theory by means of a gauge field. As discussed in the introduction, the gauge field usually turns out to be a connection between gauge choices at neighbouring points, but there is no immediate reason why this should be the case. In the first part, we formalised, in the framework of Cellular Automata (CA), the notions of gauge extension and relative gauge extension. The latter forces the gauge field to act as a connection.

(a) Evolution of F .(b) Evolution of F' .

■ **Figure 5** Evolution of F and its gauge extension F' over 9000 temporal step.

Again in Physics one usually starts from a theory featuring a global symmetry, before “making it local” through the gauge extension. Again there is no immediate reason why this should be the case. In our framework, we were able to establish a logical relation between global symmetry and relative gauge-invariance. Namely we proved that the CA that admit a relative gauge extension are exactly those that have the corresponding global symmetry. To the best of our knowledge, no continuous equivalent of that theorem exists in the literature; perhaps the discrete offers better opportunities for formalisation.

We also proved that any CA can be extended into a gauge-invariant one. Thus, gauge-invariant CA are universal. Two different constructions were provided. One construction uses the gauge field to store, at each location, the value of the gauge-transformation which the matter field has undergone at that location, thereby allowing for the action of the transformation to be counteracted. This path uses a non-relative gauge extension. Another construction puts together the fact that any CA can be made globally-symmetric [16], with the fact that any globally-symmetric CA admits a relative gauge extension. Thus, relative gauge-extended CA are universal.

Whilst the introduction of the gauge field is initially motivated by the gauge symmetry requirement, the gauge field ends up triggering new, richer behaviours as it influences the matter field. However, in order for it to mediate the interactions within the matter field, as is the case in Physics, it should be the case that the matter field also influences the gauge field – and back. In this paper, we provided a first example of a gauge-extended CA whose matter field injectively influences gauge field, whilst preserving reversibility. This was done through a general gauge extension, we leave it open whether this can be achieved through a relative gauge extension. The difficulty here is that relative gauge extensions seem to store just the minimal amount of information required for gauge-invariance, and any further influence upon the gauge field runs the risk of jeopardising that.

This difficulty can be circumvented in the quantum setting: the Quantum Cellular Automaton of [1] arises from a relative gauge extension, and yet features a gauge field which is “sourced” by the matter field. The construction directly yields a quantum simulation algorithm for one-dimensional quantum electrodynamics. This should serve us a reminder that whilst this work is theoretical, it is not merely of theoretical interest. Gauge extensions is exactly what one needs to do in order to capture physical interactions within discrete quantum models. This may lead for instance to digital quantum simulation algorithms, with improved numerical accuracy, as fundamental symmetries are preserved throughout the computation.

References

- 1 Pablo Arrighi, Cédric Bény, and Terry Farrelly. A quantum cellular automaton for one-dimensional qed. *Local proceedings of AQIS 2019. Quantum Information Processing*, 19:88, 2020. arXiv preprint. [arXiv:1903.07007](https://arxiv.org/abs/1903.07007).
- 2 Pablo Arrighi, Giuseppe Di Molfetta, and Nathanaël Eon. A gauge-invariant reversible cellular automaton. In *International Workshop on Cellular Automata and Discrete Complex Systems*, pages 1–12. Springer, 2018.
- 3 Pablo Arrighi, Giuseppe Di Molfetta, and Nathanaël Eon. Non-abelian gauge-invariant cellular automata. In *International Conference on Theory and Practice of Natural Computing*, pages 211–221. Springer, 2019.
- 4 Pablo Arrighi, Giuseppe Di Molfetta, and Nathanaël Eon. Gauge-invariance in cellular automata. *arXiv preprint*, 2020. [arXiv:2004.03656](https://arxiv.org/abs/2004.03656).
- 5 Pablo Arrighi, Nicolas Durbec, and Aurélien Emmanuel. Reversibility vs local creation/destruction. In *Proceedings of RC 2019, LLNCS*, volume 11497, pages 51–66. Springer, 2019. doi:10.1007/978-3-030-21500-2_4.
- 6 Shailesh Chandrasekharan and U-J Wiese. Quantum link models: A discrete approach to gauge theories. *Nuclear Physics B*, 492(1-2):455–471, 1997.
- 7 Péter Gács. Reliable computation with cellular automata. *Journal of Computer and System Sciences*, 32(1):15–78, 1986.
- 8 Péter Gács. Reliable cellular automata with self-organization. *Journal of Statistical Physics*, 103(1):45–267, 2001.
- 9 IM Georgescu, Sahel Ashhab, and Franco Nori. Quantum simulation. *Reviews of Modern Physics*, 86(1):153, 2014.
- 10 Pierre Guillon, Pierre-Etienne Meunier, and Guillaume Theyssier. Clandestine Simulations in Cellular Automata. 18 pages, 2010. URL: <https://hal.archives-ouvertes.fr/hal-00521624>.
- 11 Masateru Harao and Shoichi Noguchi. Fault tolerant cellular automata. *Journal of computer and system sciences*, 11(2):171–185, 1975.
- 12 W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- 13 A Yu Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003.
- 14 Chetan Nayak, Steven H Simon, Ady Stern, Michael Freedman, and Sankar Das Sarma. Non-abelian anyons and topological quantum computation. *Reviews of Modern Physics*, 80(3):1083, 2008.
- 15 Simone Notarnicola, Mario Collura, and Simone Montangero. Real-time-dynamics quantum simulation of $(1+1)$ -dimensional lattice qed with rydberg atoms. *Physical Review Research*, 2(1):013288, 2020.
- 16 Ville Salo and Ilkka Törmä. Color blind cellular automata. In *International Workshop on Cellular Automata and Discrete Complex Systems*, pages 139–154. Springer, 2013.
- 17 Pietro Silvi, Enrique Rico, Tommaso Calarco, and Simone Montangero. Lattice gauge tensor networks. *New Journal of Physics*, 16(10):103015, October 2014. doi:10.1088/1367-2630/16/10/103015.
- 18 A. Toom. Cellular automata with errors: Problems for students of probability. *Topics in Contemporary Probability and Its Applications*, pages 117–157, 1995.


A Proof of gauge-invariance for Sec. 6

In order to prove that the example illustrated in Fig. 4 is gauge-invariant, we will show that $\gamma' \circ F' = F' \circ \gamma'$ for any $\gamma' \in \Gamma'$. It is sufficient to prove this locally, we do so using the notations of the figure and we denote by f' and g' the local application of the evolution and a gauge-transformation:


$$\begin{aligned} f' \circ g'(c_1^l, a_1, a_2, c_2^r) &= \begin{pmatrix} c_1^l + c_2^r + a_1 + k_1, \\ a_2 + c_1^l - k_2, \\ a_1 + a_2 + c_2^r - k_1 - k_2, \\ c_1^l - c_2^r + a_2 + k_2 \end{pmatrix} \\ &= g' \circ f'(c_1^l, a_1, a_2, c_2^r) \end{aligned}$$

Therefore F' is Γ' -gauge-invariant.

Equivalence Testing of Weighted Automata over Partially Commutative Monoids*

V. Arvind 


Institute of Mathematical Sciences (HBNI), Chennai, India

Abhranil Chatterjee 

Institute of Mathematical Sciences (HBNI), Chennai, India

Rajit Datta 

Chennai Mathematical Institute, India

Partha Mukhopadhyay 

Chennai Mathematical Institute, India

Abstract

Motivated by equivalence testing of k -tape automata, we study the *equivalence* testing of weighted automata in the more general setting, over partially commutative monoids (in short, pc monoids), and show efficient algorithms in some special cases, exploiting the structure of the underlying non-commutation graph of the monoid.

Specifically, if the edge clique cover number of the non-commutation graph of the pc monoid is a constant, we obtain a deterministic quasi-polynomial time algorithm for equivalence testing. As a corollary, we obtain the first deterministic quasi-polynomial time algorithms for equivalence testing of k -tape weighted automata and for equivalence testing of deterministic k -tape automata for constant k . Prior to this, the best complexity upper bound for these k -tape automata problems were randomized polynomial-time, shown by Worrell [24]. Finding a polynomial-time deterministic algorithm for equivalence testing of deterministic k -tape automata for constant k has been open for several years [13] and our results make progress.

We also consider pc monoids for which the non-commutation graphs have an edge cover consisting of at most k cliques and star graphs for any constant k . We obtain a randomized polynomial-time algorithm for equivalence testing of weighted automata over such monoids.

Our results are obtained by designing efficient zero-testing algorithms for weighted automata over such pc monoids.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation

Keywords and phrases Weighted Automata, Automata Equivalence, Partially Commutative Monoid

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.10

Acknowledgements We thank the anonymous reviewers for their helpful feedback.

1 Introduction

Testing the equivalence of two multi-tape finite automata is a fundamental problem in automata theory. For a k -tape automaton, we denote the mutually disjoint alphabets for the k tapes by $\Sigma_1, \dots, \Sigma_k$. The automaton accepts a subset of the product monoid $\Sigma_1^* \times \dots \times \Sigma_k^*$. Two multi-tape automata are *equivalent* if they accept the same subset.

* A preliminary version of this work was presented at HIGHLIGHTS of Logic, Games and Automata 2020.



Equivalence testing of multi-tape *non-deterministic* automata is undecidable [14]. The problem was shown to be decidable for 2-tape *deterministic* automata independently by Bird [5] and Valiant [22]. Subsequently, an exponential upper bound was shown for it [3]. Eventually, a polynomial-time algorithm was obtained by Friedman and Greibach [13] and the authors conjectured that equivalence testing of deterministic k -tape automata for any constant k is in polynomial time.

A closely related problem is testing the *multiplicity equivalence* of non-deterministic multi-tape automata. The multiplicity equivalence testing problem is to decide whether for each tuple in the product monoid $\Sigma_1^* \times \dots \times \Sigma_k^*$, the number of accepting paths in the two input automata is the same. Since a deterministic automaton has at most one accepting path for each input word, the equivalence of deterministic k -tape automata coincides with multiplicity equivalence. More generally, equivalence testing for *weighted automata* (over the underlying field or ring of coefficients) is to decide if the coefficient of each word (i.e. the total sum of weights of each accepting path) is the same for the two given automata. For the weighted case, equivalence testing is in deterministic polynomial time for one-tape automata [19, 21]. Equivalence testing of k -tape weighted automata was shown *decidable* by Harju and Karhumäki [15] using the theory of free groups¹. An improved complexity-theoretic upper bound remained elusive for k -tape multiplicity equivalence testing, until recently Worrell [24] obtained a *randomized* polynomial-time algorithm for testing the equivalence of k -tape weighted automata (and equivalence testing of deterministic k -tape automata) for any constant k . Worrell takes a different approach via Polynomial Identity Testing (PIT). In [24], Worrell asked if the equivalence testing problem for k -tape weighted automata can be solved in *deterministic* polynomial time, for constant k .

This Paper. Building on Worrell’s results [24] and exploiting further the connections between weighted automata equivalence and polynomial identity testing, we show that equivalence testing of two k -tape weighted automata is in *deterministic* quasi-polynomial time. This immediately yields the first deterministic quasi-polynomial time algorithm for equivalence testing of deterministic k -tape automata.

Our approach solves a more general problem in the setting of *partially commutative monoids*. To motivate this, let us consider k -tape weighted automata in this setting. The *product monoid* $M = \Sigma_1^* \times \dots \times \Sigma_k^*$ associated with k -tape automata is a *partially commutative monoid* (henceforth, *pc monoid*), in the sense that any two variables $x \in \Sigma_i, y \in \Sigma_j, i \neq j$ commute with each other². Variables in the same tape alphabet Σ_i are mutually non-commuting. We associate a *non-commutation graph* G_M with M to describe the non-commutation relations: (x, y) is an edge if and only if x and y do not commute. If there is no edge (x, y) in G_M , the words xy and yx are equivalent as the variables x and y commute. The words over any pc monoid are defined with respect to the equivalence relation induced by the non-commutation graph of the pc monoid. The notion of words and their equivalence over a pc monoid is formally explained in Section 3. For the k -tape case, the non-commutation graph G_M is a union of k disjoint cliques: its vertex set is $\Sigma_1 \cup \dots \cup \Sigma_k$ and G_M is the union of k disjoint cliques, induced by each Σ_i .

More generally, we obtain an equivalence testing algorithm for weighted automata over any pc monoid whose non-commutation graph has a constant-size edge clique cover (*not necessarily* disjoint) with a constant number of isolated vertices. Recall that the edge clique

¹ This also shows the decidability of equivalence problem for deterministic multi-tape automata.

² These are sometimes also called as free partially commutative monoids.

cover of a graph is a collection of subgraphs where each subgraph is a clique and each edge of the graph is contained in at least one of the subgraphs. The size of the edge clique cover is the number of cliques in it.

The isolated vertices can be thought of as a part of the edge clique cover by adding a new vertex (variable) for each isolated vertex and introducing a matching edge between them. Henceforth, we will not worry about the isolated vertices separately and consider them as part of the edge clique cover. We call such monoids as *k-clique monoids* where the edge clique cover size is bounded by k .

► **Remark 1.1.** Since two weighted automata, A and B are equivalent if and only if their difference $C = A - B$ is a weighted automaton equivalent to zero (formally explained in Section 2), we can describe the results in terms of zero-testing of a weighted automaton.³

In this paper, the field \mathbb{F} from which the weights of automata are taken is an infinite field. For computational implementation, we assume that the field arithmetic can be performed efficiently (for example, \mathbb{F} could be the field of rational numbers). Also, throughout the paper the size of an automaton refers to the number of states.

► **Theorem 1.2.** *Let A be an input \mathbb{F} -weighted automaton of size s over a pc monoid M such that its non-commutation graph G_M has an edge clique cover of size k . Then, the zero-testing of A has a deterministic $(nks)^{O(k^2 \log ns)}$ time algorithm. Here n is the size of the alphabet of M , and the edge clique cover is given as part of the input.*

It is interesting to note that the decidability of the equivalence problem over partially commutative monoids is already studied [23]. As an immediate corollary, the above theorem yields a deterministic quasi-polynomial time algorithm for equivalence testing of k -tape weighted automata (also for equivalence testing of deterministic k -tape automata). Notice that, for the k -tape case, the edge clique cover of size k is also part of the input since for each $1 \leq i \leq k$, the i^{th} tape alphabet Σ_i is explicitly given and it induces a clique.

► **Corollary 1.3.** *The equivalence testing problem for k -tape weighted automata and deterministic k -tape automata can be solved in deterministic quasi-polynomial time for constant k .*

Next, we consider equivalence testing over more general pc monoids M .

Given a graph $G = (X, E)$, a collection of k graphs $\{G_i = (X_i, E_i)\}_{i=1}^k$ such that $X = \cup_{i=1}^k X_i$ and $E = \cup_{i=1}^k E_i$ is called a *k-covering* of G . It seems natural to investigate whether there are covers other than just edge clique cover for which one can obtain efficient equivalence test.

We say M is a *k-monoid* if its non-commutation graph G_M has a 2-covering $\{G_1, G_2\}$ such that, for some $k' \leq k$, G_1 has an edge clique cover of size at most k' and G_2 has a vertex cover of size at most $k - k'$ (hence the edges of G_2 can be covered by $k - k'$ many star graphs). We show that equivalence testing over k -monoids has a randomized polynomial-time algorithm for constant k . This result can be seen as a generalization of Worrell's result [24].

► **Theorem 1.4.** *Let A be an input \mathbb{F} -weighted automaton of size s over a k -monoid M . Then the zero-testing of A can be decided in randomized $(ns)^{O(k)}$ time. Here n is the size of the alphabet of M .*

³ The difference C of two weighted automata A and B means the weight of each word w in C is the difference between the weights of w in A and B .

10:4 Equivalence Testing of Weighted Automata

► **Remark 1.5.** What is the complexity of equivalence testing for weighted automata over an arbitrary pc monoid? The non-commutation graph G_M of any pc monoid M over the alphabet X trivially has an edge clique cover of size bounded by $\binom{|X|}{2}$. Hence, the above results would only give an exponential-time algorithm. Note that if G_M has an *induced matching*⁴ of size more than k then M is not a k -monoid. Call M a *matching monoid* if G_M is a perfect matching. It follows from Lemma 3.3, shown in Section 3, that equivalence testing over arbitrary pc monoids is deterministic polynomial-time reducible to equivalence testing over matching monoids. Thus, the complexity of zero-testing of \mathbb{F} -weighted automata over matching monoids is essentially the most general case. We also note that Worrell has shown that the evaluation problem for multi-tape automata is $\#P$ -complete if the number of tapes is not fixed [24, Proposition 3].

Various automata-theoretic problems have been studied in the setting of pc monoids. For example, pc monoids have found applications in modeling the behavior of concurrent systems [16]. Droste and Gastin [10] have studied the relation between recognizability and rationality over pc monoids.

Proofs Overview. Our proof is inspired by Worrell’s key insight [24] that the k -tape automata equivalence problem can be reduced to a suitable instance of polynomial identity testing problem over partially commuting variables. Worrell’s algorithm is randomized. In contrast, since we are considering automata over general pc monoids and we aim to design efficient deterministic algorithms, we require additional ideas. First, we suitably apply a classical algebraic framework to transfer the zero-testing problem over general pc monoids to pc monoids whose non-commutation graphs are a disjoint union of cliques [6, 8]. This allows us to generalize a zero-testing criteria for weighted automata over standard noncommutative setting [11, Cor. 8.3] to the setting of *general pc monoids*. The generalization states that any nonzero weighted automata of size s over any pc monoid must have a non-zero word within the length $\text{poly}(s, n)$ where n is the alphabet size. This allows us to reduce zero-testing of weighted automata to an instance of polynomial identity testing over pc monoids, where these polynomials are computable by small algebraic branching programs (ABPs) over pc monoids. Over noncommutative variables, ABPs are well-studied in arithmetic circuit complexity [17]. It turns out that we can solve the identity testing problem for ABPs over k -clique monoids in deterministic quasi-polynomial time by suitably adapting a black-box polynomial identity test for noncommutative algebraic branching programs based on a quasi-polynomial size hitting set construction [12]. Our algorithm recursively builds on this construction, ensuring that the resulting hitting set remains of quasi-polynomial size.

The proof of Theorem 1.4 is along similar lines. First, we obtain a randomized polynomial-time identity testing algorithm over pc monoids whose non-commutation graph has a k -vertex cover for constant k . This algorithm itself uses ideas from automata theory. Then a composition lemma yields an identity testing algorithm over k -monoids.

The paper is organized as follows. In Section 2, we provide the necessary background. We prove a zero testing criteria for automata over pc monoids in Section 3. Theorem 1.2 is presented in Section 4, and Theorem 1.4 in Section 5. Some proof details are in the appendix.

⁴ An induced matching is a matching that includes every edge connecting any two vertices in the subset as an induced subgraph.

2 Preliminaries

We recall basic definitions and results, mainly from automata theory and arithmetic circuit complexity, and define notations used in the paper.

Notation. Let \mathbb{F} be an infinite field. $\text{Mat}_t(\mathbb{F})$ denotes the ring of $t \times t$ matrices over \mathbb{F} . For matrices A and B of sizes $m \times n$ and $p \times q$ respectively, their tensor (Kronecker) product $A \otimes B$ is defined as the block matrix $(a_{ij}B)_{1 \leq i \leq m, 1 \leq j \leq n}$, and the dimension of $A \otimes B$ is $pm \times qn$. Given bases $\{v_i\}_{1 \leq i \leq \dim(V)}$ and $\{w_j\}_{1 \leq j \leq \dim(W)}$ for the vector spaces V and W , the vector space $V \otimes W$ is the tensor product space with a basis $\{v_i \otimes w_j\}_{1 \leq i \leq \dim(V), 1 \leq j \leq \dim(W)}$.

For a series (resp. polynomial) S and a word (resp. monomial) w , let $[w]S$ denote the coefficient of w in the series S (resp. polynomial). In this paper, we consider weighted automata over a field \mathbb{F} and alphabet (or variables) $X = \{x_1, \dots, x_n\}$.

Arithmetic Circuit Complexity. An *algebraic branching program* (ABP) is a layered directed acyclic graph with one in-degree-0 vertex called *source*, and one out-degree-0 vertex called *sink*. Its vertex set is partitioned into layers $0, 1, \dots, d$, with directed edges only between adjacent layers (i to $i + 1$). The source and the sink are in layers zero and d , respectively. Each edge is labeled by a linear form over \mathbb{F} in variables $X = \{x_1, \dots, x_n\}$. The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the product of linear forms that label the edges of the path. The maximum number of nodes in any layer is called the width of the algebraic branching program. The size of the branching program is taken to be the total number of nodes.

Equivalently, the computation of an algebraic branching program can be defined via the iterated matrix product $\lambda^T M_1 M_2 \cdots M_d \mu$, where λ, μ are vectors in \mathbb{F}^w and each M_i is a $w \times w$ matrix whose entries are affine linear forms over X . Here w corresponds to the ABP width and $d + 1$ corresponds to the number of layers in the ABP.

If X is a set of non-commuting variables then the ABP is a noncommutative algebraic branching program (e.g., see [17]).

Let $S \subset \mathbb{F}\langle X \rangle$ be a subset of polynomials in the noncommutative polynomial ring $\mathbb{F}\langle X \rangle$. A mapping $\mathbf{v} : X \rightarrow \text{Mat}_t(\mathbb{F})$ from variables to $t \times t$ matrices, it defines an *evaluation map* defined for any polynomial $f \in \mathbb{F}\langle X \rangle$ as $\mathbf{v}(f) = f(\mathbf{v}(x_1), \dots, \mathbf{v}(x_n))$. A collection H of such evaluation maps is a *hitting set* for S , if for every nonzero f in S , there is an evaluation $\mathbf{v} \in H$ such that $\mathbf{v}(f) \neq 0$.

Let $S_{n,d,s}$ denote the set of noncommutative polynomials in $\mathbb{F}\langle X \rangle$ (where $n = |X|$) that have algebraic branching programs of size s and d layers. Forbes and Shpilka [12] have given a quasi-polynomial size hitting set $H_{n,d,s}$ computable in quasi-polynomial time for $S_{n,d,s}$ that can. Moreover, the matrix tuples in $H_{n,d,s}$ are $d + 1$ dimensional.

► **Theorem 2.1** ([12, Theorem 1.8]). *For all $s, d, n \in \mathbb{N}$, if $|\mathbb{F}| \geq \text{poly}(d, n, s)$ then there is a hitting set $H_{n,d,s}$ for $S_{n,d,s}$. Further $|H_{n,d,s}| \leq (sdn)^{O(\log d)}$ and $H_{n,d,s}$ is computable in deterministic time $(sdn)^{O(\log d)}$.*

Automata Theory. We recall some basic algebraic automata theory from the Berstel-Reutenauer book [4].

Let \mathbb{F} be a field⁵ and X be an alphabet. A \mathbb{F} -weighted automaton⁶ A over X has a finite set of states Q . There is a weight function $E : Q \times X \times Q \rightarrow \mathbb{F}$ that assigns a weight to each transition. The number of states, $|Q|$ is the *size* of the automaton. A path is a sequence

⁵ In general \mathbb{F} can be a semiring, but for our purpose it suffices to consider fields.

⁶ Sometime called nondeterministic weighted automata in the literature.

10:6 Equivalence Testing of Weighted Automata

of edges: $(q_0, x_1, q_1)(q_1, x_2, q_2) \cdots (q_{t-1}, x_t, q_t)$. The weight of the path is the product of the weights of the edges. For each word $w = x_1 x_2 \cdots x_t \in X^*$, the coefficient of w , $[w]S$ is the total contribution of all the paths between a start and accepting state for the word w , which is an element of \mathbb{F} . This defines a formal series $S = \sum_{w \in X^*} [w]S \cdot w$ which is an element of the *formal power series ring* $\mathbb{F}\langle\langle X \rangle\rangle$. We say that S is the formal series *recognized* by the (weighted) automaton A .

Multi-tape automata. Next, we briefly explain weighted multi-tape automata defined in terms of pc monoids. Let M be the pc monoid over variables $X = X_1 \cup \cdots \cup X_k$ defined as follows: the variables in each X_i are non-commuting, but for all $i \neq j$ and any $x \in X_i, y \in X_j$ we have $xy = yx$. As defined already, the transition function E is a mapping $Q \times X \times Q \rightarrow \mathbb{F}$. A path is a sequence of edges: $(q_0, x_1, q_1)(q_1, x_2, q_2) \cdots (q_{t-1}, x_t, q_t)$ where each $x_i \in X_j$ for some j . The label of the run is $m = x_1 x_2 \cdots x_t$ in the pc monoid M , and $[m]A$ is the total contribution of all the runs between start and accepting states having the label equivalent to m .

An automaton is *deterministic* if the set of states can be partitioned as $Q = Q^{(1)} \sqcup \cdots \sqcup Q^{(k)}$, where states in $Q^{(i)}$ read input only from the i^{th} tape alphabet X_i , and each state has a single transition for every input variable. Thus, a deterministic automaton has at most one accepting path for each input $m \in M$.

Now we explain how equivalence testing of weighted automata is polynomial-time reducible to zero testing of weighted automata. Let A and B be \mathbb{F} -weighted automata over the alphabet X . The transition matrices N_A and N_B are defined as follows: $N_A[i, j] = \sum_{x \in X} E_A(q_i, x, q_j) \cdot x$. (N_B is defined similarly)⁷. Let the series computed by A and B be $\lambda_A^T \cdot \sum_{i \geq 0} N_A^i \cdot \mu_A$ and $\lambda_B^T \cdot \sum_{i \geq 0} N_B^i \cdot \mu_B$, respectively. Here $\lambda_A, \mu_A, \lambda_B, \mu_B$ are column scalar vectors. Define the weighted automaton C with transition matrix N_C and the scalar vectors λ_C, μ_C as follows:

$$\lambda_C = \begin{bmatrix} \lambda_A \\ \lambda_B \end{bmatrix}, \quad N_C = \begin{bmatrix} N_A & 0 \\ 0 & N_B \end{bmatrix}, \quad \mu_C = \begin{bmatrix} \mu_A \\ -\mu_B \end{bmatrix}.$$

We state an easy fact also used in [24].

► **Fact 1.** *A and B are equivalent if and only if C is a zero automaton.*

3 A Zero Testing Criteria Over Partially Commutative Monoids

A basic result in algebraic automata theory, says that an \mathbb{F} -weighted automaton A of size s represents a nonzero series in $\mathbb{F}\langle\langle X \rangle\rangle$ if and only if there is a word $w \in X^*$ of length at most $s - 1$, such that $[w]S$ is nonzero. It has a simple linear algebraic proof [11, Corollary 8.3, Page 145]⁸.

In this section, we prove a theorem similar in spirit over general pc monoids.

Pc monoids and partitioned pc monoids. Let X be a finite alphabet (equivalently, variable set). Formally, a pc monoid M over X is a pair $M = (X^*, I)$ where $I \subseteq X \times X$ be such that $(x_1, x_2) \in I$ if and only if $x_1 x_2 = x_2 x_1$. I is reflexive and symmetric. Let \tilde{I} be the *congruence* generated from I by transitive closure. The monoid elements are defined as the congruence classes \tilde{m} for $m \in X^*$. In other words, M is a factor monoid of X^* generated by \tilde{I} . The *non-commutation graph* $G_M = (X, E)$ of M is a simple undirected graph such that $(x_1, x_2) \in E$ if and only if $(x_1, x_2) \notin I$.

⁷ Here q_1, q_2, \dots be the enumeration of the states of A (and similarly for B).

⁸ This result is generally attributed to Schützenberger.

A pc monoid M over alphabet (i.e. variable set) X is a k -partitioned pc monoid if its non-commutation graph G_M has a k -covering $\{G_i\}_{i=1}^k$ such that the subgraphs G_i are pairwise vertex disjoint. Given any pc monoid M with a k -covering, we can associate a k -partitioned pc monoid M' with it, such that M is isomorphic to a submonoid of M' , as follows.

Suppose $G_M = (X, E)$ has a k -covering $\{G_i\}_{i=1}^k$, where $G_i = (X_i, E_i)$. Let $\hat{X} = \{x_{ti} \mid 1 \leq t \leq n, 1 \leq i \leq k\}$ be kn new variables. Here $|X| = n$ and $X = x_1, \dots, x_n$. Let $G'_i = (X'_i, E'_i)$ be a copy of G_i obtained by replacing the vertex $x_t \in X_i$ by its i^{th} copy x_{ti} , such that (x_{ti}, x_{si}) is an edge in G'_i if and only if (x_t, x_s) is an edge in G_i . Let G' denote the disjoint union graph $G' = G'_1 \sqcup G'_2 \sqcup \dots \sqcup G'_k$, and M' be the pc monoid whose non-commutation graph is $G' = (X', E')$. Clearly, M' is a k -partitioned pc monoid, defined by M and its given k -covering.

As an \mathbb{F} -algebra, we note that $\mathbb{F}\langle M' \rangle$ is isomorphic to the tensor product of the \mathbb{F} -algebras $\mathbb{F}\langle M'_1 \rangle \otimes \dots \otimes \mathbb{F}\langle M'_k \rangle$ where M'_i is the pc monoid defined by G'_i .

The following simple observation, which shows that the pc monoid M is isomorphic to a submonoid of M' , is well-known [6, 8, 9].

► **Lemma 3.1.** *Let $\psi : \mathbb{F}\langle M \rangle \rightarrow \mathbb{F}\langle M' \rangle$ be the map such that $\psi(m) = m_1 \otimes m_2 \otimes \dots \otimes m_k$ for any monomial m in M and extend by linearity. Here, for $1 \leq i \leq k$, the monomial m_i is obtained from m (by dropping the letters of m not in X_i) and replacing each occurrence $x_t \in X_i$ by the variable x_{ti} , $1 \leq t \leq n$. Then, ψ is an injective homomorphism.*

► **Remark 3.2.** We include a self-contained proof in the appendix for completeness, in our notation.

By Lemma 3.1, zero testing for weighted automata over pc monoids is reducible to zero-testing of weighted automata over partitioned pc monoids in deterministic polynomial time. More formally, we show the following.

► **Lemma 3.3.** *Let A be the given \mathbb{F} -weighted automaton of size s over a pc monoid M for which the non-commutation graph G_M has k -covering $\{G_i = (X_i, E_i)\}_{i=1}^k$. Then zero testing of A is reducible to the zero testing of another \mathbb{F} -weighted automaton B over the associated k -partitioned pc monoid M' in deterministic polynomial time. Moreover, the size of the automaton B is $O(ns^2k)$.*

Proof. The automaton B is simply obtained by applying the map ψ on the variables in M . For a variable x_t , let $J_t \subseteq \{1, 2, \dots, k\}$ be the set of indices such that, $i \in J_t$ if and only if $x_t \in X_i$. Then $\psi(x_t) = \eta_{i_1} \otimes \dots \otimes \eta_{i_{|J_t|}}$ where $i_1 < i_2 < \dots < i_{|J_t|}$ and for each j , $i_j \in J_t$, $\eta_{i_j} = x_{ti_j}$. Now for each $q_0, q_k \in Q$ such that $(q_0, x_t, q_k) \in E$ ⁹ and $wt(q_0, x_t, q_k) = \alpha \in \mathbb{F}$, we introduce new states $q_1, \dots, q_{|J_t|-1}$ and for each $j \leq |J_t| - 1$, add the edge $e_j = (q_{j-1}, \eta_{i_j}, q_j)$ in E and $wt(e_1) = \alpha$ and for other newly added edges the weight is 1. Since the number of edges in A is $O(ns^2)$, it is easy to see the number of nodes in B is $O(ns^2k)$. The fact that A computes the zero series if and only if B computes the zero series, follows from Lemma 3.1 and in particular from the fact that ψ is injective on the set of monomials. ◀

Worrell has already proved that the zero-testing of weighted automata over partitioned monoids whose non-commutation graphs are union of disjoint cliques, can be reduced to the identity testing of noncommutative ABPs [24]. We restate a proposition from Worrell's paper in our framework.

⁹ Here for simplicity of notation, we have used q_0, q_k to represent an arbitrary pair such that there is a transition between them, and q_0 is not necessarily the initial state.

► **Proposition 3.4** (Adaptation of Proposition 5 of [24]). *Let A be a given \mathbb{F} -weighted automaton of size s over a partitioned pc monoid M computing a series S . Moreover the non-commutation graph G_M is the disjoint union of k cliques. Let N be the transition matrix of A . Then S is the zero series if and only if the ABPs $\lambda^T N^\ell \mu = 0$ for each $0 \leq \ell \leq s - 1$, where λ, μ are vectors in \mathbb{F}^s .*

Combining Lemma 3.3 and Proposition 3.4, we obtain the following generalization of [11, Corollary 8.3] over arbitrary pc monoids which may be of independent interest.

► **Theorem 3.5.** *Let A be a given \mathbb{F} -weighted automaton of size s over any pc monoid M representing a series S . Then S is a nonzero series if and only if there exists a word $w \in X^*$ such that $[w]S$ is nonzero where the length of w is bounded by $O(n^3 s^2)$.*

Proof. Observe that the non-commutation graph G_M has a trivial edge clique cover of size $\leq n^2$ where n is the size of the alphabet. Then we apply Lemma 3.3 to conclude that S is a zero series if and only if the series S' computed by the \mathbb{F} -weighted automaton B over the associated partitioned pc monoid (whose non-commutation graph is a disjoint union of cliques) is zero. The size s' of B is bounded by $O(n^3 s^2)$. Now we use Proposition 3.4 to see that S' is identically zero if and only if the ABPs $\lambda^T N^\ell \mu = 0$ for each $0 \leq \ell \leq s' - 1$ are identically zero where N is the transition matrix of B . Now notice that under the image of ψ map, the length of any word can only increase. In other words, for any word $w : |\psi(w)| \geq |w|$. It follows that $(S' = \psi(S))^{\leq s'-1}$ is a nonzero polynomial where S' is the part of $\psi(S)$ of degree at most $s' - 1$. Since ψ is injective, it must be the case that $S^{\leq s'-1}$ is also a nonzero polynomial, and the theorem follows. ◀

4 Deterministic Zero Testing of Weighted Automata Over k -Clique Monoids

In this section, we show that zero testing for weighted automata over k -clique monoids for constant k is in deterministic quasi-polynomial time. In fact, by Lemma 3.3 and Proposition 3.4, the zero testing problem reduces to the polynomial identity testing of ABPs over partitioned pc monoids whose non-commutation graph is a disjoint union of k cliques. Thus, in order to prove Theorem 1.2 it suffices to design an efficient identity testing algorithm for ABPs computing polynomials in $\mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$, where k is a constant and the variable sets $X_j = \{x_{ij}\}_{1 \leq i \leq n}$ are of size n each and pairwise disjoint.

Evaluation over algebras. For a polynomial $f \in \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ and a k -tuple of \mathbb{F} -algebras $\mathbf{A} = (A_1, \dots, A_k)$, an *evaluation* of f in \mathbf{A} is given by a k -tuple of maps $\mathbf{v} = (v_1, v_2, \dots, v_k)$, where $v_i : X_i \rightarrow A_i$. We can extend it to the map $\mathbf{v} : \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle \rightarrow A_1 \otimes \cdots \otimes A_k$ as follows: For any monomial $m = m_1 \otimes \cdots \otimes m_k$ where $m_i \in X_i^*$, let $\mathbf{v}(m) = v_1(m_1) \otimes \cdots \otimes v_k(m_k)$. In particular, for each $x \in X_j$ let $\mathbf{v}(x) = 1_1 \otimes \cdots \otimes v_j(x) \otimes \cdots \otimes 1_k$ where 1_j is the multiplicative identity of A_j . We can now extend \mathbf{v} by linearity to all polynomials in the domain $\mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$.

Next, we define a *partial evaluation* of $f \in \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ in \mathbf{A} . Let $k' < k$ and $\hat{\mathbf{A}} = (A_1, \dots, A_{k'})$ be a k' -tuple of \mathbb{F} -algebras. A partial evaluation of $\mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ in $\hat{\mathbf{A}}$ is given by a k' -tuple of maps $\hat{\mathbf{v}} = (v_1, \dots, v_{k'})$, where $v_i : X_i \rightarrow A_i$. Now, we can define $\hat{\mathbf{v}} : \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle \rightarrow A_1 \otimes \cdots \otimes A_{k'} \otimes \mathbb{F}\langle X_{k'+1} \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ as follows. For a monomial $m = (m_1 \otimes \cdots \otimes m_k)$, $m_i \in X_i^*$, we let $\hat{\mathbf{v}}(m) = v_1(m_1) \otimes \cdots \otimes v_{k'}(m_{k'}) \otimes m_{k'+1} \otimes \cdots \otimes m_k$. By linearity, the partial evaluation $\hat{\mathbf{v}}$ is defined for any $f \in \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ where $\hat{\mathbf{v}}$ takes values in $A_1 \otimes \cdots \otimes A_{k'} \otimes \mathbb{F}\langle X_{k'+1} \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$.

Although it is implicit, we formally recall that when we consider ABPs over $\mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ the linear forms are defined over tensors of the form $1 \otimes \cdots \otimes x_{ij} \otimes \cdots \otimes 1$. These tensors play the role of an individual variable in the tensor product structure.

Some more notation. Let $S_{k,n,d,s}$ denote the set of all polynomials in $\mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ computed by ABPs of size s and layers $0, 1, \dots, d$, and $n = |X_i|$ for each $1 \leq i \leq k$. Following the notation in Theorem 2.1, we will denote by $\mathcal{H}_{k,n,d,s}$ the hitting set that we will construct for $S_{k,n,d,s}$. That is, $\mathcal{H}_{k,n,d,s}$ is a collection of evaluations in the ring of square matrices $\mathbf{v} = (v_1, \dots, v_k)$, such that for any nonzero polynomial $f \in S_{k,n,d,s}$ there is an evaluation $\mathbf{v} = (v_1, \dots, v_k) \in \mathcal{H}_{k,n,d,s}$ such that $\mathbf{v}(f)$ is a nonzero matrix. Recall from Theorem 2.1 that a quasi-polynomial size hitting set $\mathcal{H}_{1,n,d,s}$ for $S_{1,n,d,s}$ can be explicitly constructed. In the next lemma we describe an efficient bootstrapped construction of the hitting set $\mathcal{H}_{k,n,d,s}$ for the set $S_{k,n,d,s}$ of polynomials, from the hitting set $\mathcal{H}_{1,n,d,s}$.

► **Lemma 4.1.** *There is a set of evaluation maps $\mathcal{H}_{k,n,d,s} = \{(v_1, \dots, v_k) : v_i \in \mathcal{H}_{1,n,d,s_k}\}$ where $s_k = s(d+1)^{(k-1)}$ such that, for $i \in [k]$, we have $v_i : X_i \rightarrow \mathcal{M}_{d+1}(\mathbb{F})$, and $\mathcal{H}_{k,n,d,s}$ is a hitting set for the class of polynomials $S_{k,n,d,s}$. Moreover, the size of the set is at most $(nskd)^{O(k^2 \log d)}$, and it can be constructed in deterministic $(nskd)^{O(k^2 \log d)}$ time.*

The above lemma yields the identity test: we only need to evaluate the input polynomial on each point of the hitting set and check whether it is nonzero.

Before presenting the proof, we discuss two important ingredients. A polynomial f in $\mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ can be written as $f = \sum_{m \in X_k^*} f_m \otimes m$ where each m is a monomial over variables X_k and $f_m \in \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_{k-1} \rangle$. Given that f has a small ABP, we first show that each polynomial f_m also has a small ABP.

► **Lemma 4.2.** *For each $f \in S_{k,n,d,s}$ and $m \in X_k^*$, the polynomial $f_m \in \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_{k-1} \rangle$ has an ABP of size $s(d+1)$ and $d+1$ layers.*

Proof. Suppose $f \in \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$ has an ABP B of size s and d layers, and the monomial $m = x_{i_1 k} x_{i_2 k} \cdots x_{i_\ell k}$ where some of the indices could be repeated. We will construct an ABP of size $s(d+1)$ for the polynomial f_m . First, we identify each variable $1 \otimes \cdots \otimes x_{ij} \otimes \cdots \otimes 1$ as x_{ij} and construct the following ABP B' from B :

For every node u in the ABP B , we have nodes $(u, i), 0 \leq i \leq \ell$ in the ABP B' . We now describe the edges of B' and the edge labels. In the ABP B , let (u, v) be an edge, where u is in layer j and v is in layer $j+1$, for some $j \leq d-1$. We can write the linear form labeling (u, v) as a sum $L_1 + L_2$, where L_1 is an affine linear form in variables from $X \setminus X_k$, and L_2 is a homogeneous linear form in variables from X_k .

For $0 \leq r \leq \ell-1$: we put an edge from (u, r) to (v, r) with label L_1 . For $0 \leq r \leq \ell-1$: we put an edge from (u, r) to $(v, r+1)$ with edge label $\alpha \cdot x_{i_{r+1} k}$ if the coefficient of $x_{i_{r+1} k}$ in L_2 is $\alpha \neq 0$. If s and t are the source and sink nodes of the ABP B , we designate $(s, 0)$ and (t, ℓ) as the source and sink nodes of the ABP B' .

It is evident from the construction that the ABP B' has at most $s(d+1)$ many nodes. Furthermore, the only nonzero monomials in the polynomial computed by B' are of the form $m' \otimes m$, where m' is a monomial over the letters $X \setminus X_k$, and the coefficient of $m' \otimes m$ is the same as its coefficient in polynomial f . It follows, that B' computes the polynomial $f_m \otimes m$, and we can obtain an ABP for f_m by setting to 1 all the variables occurring in m . This completes the proof. ◀

For a polynomial f in $\mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_k \rangle$, consider a partial evaluation $\mathbf{v} = (v_1, \dots, v_{k-1})$ such that each $v_i : X_i \rightarrow \mathcal{M}_{t_i}(\mathbb{F})$. The evaluation $\mathbf{v}(f)$ is a $T \times T$ matrix with entries from $\mathbb{F}\langle X_k \rangle$, where $T = t_1 t_2 \cdots t_{k-1}$.

10:10 Equivalence Testing of Weighted Automata

► **Lemma 4.3.** *For each $p, q \in [T]$ and $f \in S_{k,n,d,s}$, the $(p, q)^{th}$ entry of $\mathbf{v}(f)$ can be computed by an ABP of size sT and $d + 1$ layers.*

The proof is routine and given in the appendix. Now we are ready to prove Lemma 4.1.

Proof of Lemma 4.1. The proof is by induction on k . For the base case $k = 1$ the hitting set $\mathcal{H}_{1,n,d,s}$ of Theorem 2.1 suffices. We can write each nonzero $f \in S_{k,n,d,s}$ as $f = \sum_{m \in X_k^*} f_m \otimes m$, where $m \in X_k^*$ and $f_m \in \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_{k-1} \rangle$. Since $f \neq 0$ we have $f_m \neq 0$ for some $m \in X_k^*$. By Lemma 4.2, for each $m \in X_k^*$ the polynomial $f_m \in \mathbb{F}\langle X_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle X_{k-1} \rangle$ has an ABP of size $s(d + 1)$. Let $s' = s(d + 1)$.

By induction hypothesis, f_m is nonzero on some point in the set $\{(v_1, v_2, \dots, v_{k-1}) \mid v_i \in \mathcal{H}_{1,n,d,s'_{k-1}}\}$ where $s'_{k-1} = s'(d + 1)^{k-2} = s(d + 1)^{k-1}$. Hence, there is an evaluation $\mathbf{v}' \in \mathcal{H}_{k-1,n,d,s'}$ such that $\mathbf{v}'(f_m)$ is a nonzero matrix of dimension $(d + 1)^{k-1}$. Interpreting \mathbf{v}' as a *partial evaluation* for f , we observe that $\mathbf{v}'(f)$ is a $(d + 1)^{k-1} \times (d + 1)^{k-1}$ matrix with entries from $\mathbb{F}\langle X_k \rangle$. Since $\mathbf{v}'(f_m) \neq 0$, it follows that some $(p, q)^{th}$ entry of $\mathbf{v}'(f)$ is a nonzero polynomial $g \in \mathbb{F}\langle X_k \rangle$. By Lemma 4.3, each entry of $\mathbf{v}'(f)$ has an ABP of size $s(d + 1)^{k-1}$. In particular, $g \in S_{1,n,d,s(d+1)^{k-1}}$ and it follows from Theorem 2.1 that there is an evaluation \mathbf{v}'' in $\mathcal{H}_{1,n,d,s(d+1)^{k-1}}$ such that $\mathbf{v}''(g)$ is a nonzero matrix of dimension $(d + 1) \times (d + 1)$.

Thus, for the combined evaluation map $\mathbf{v} = (\mathbf{v}', \mathbf{v}'')$, $\mathbf{v}(f)$ is a nonzero matrix of dimension $(d + 1)^k \times (d + 1)^k$. Define $\mathcal{H}_{k,n,d,s} = \{(v_1, \dots, v_k) : v_i \in \mathcal{H}_{1,n,d,s_k}\}$, where $s_k = s(d + 1)^{k-1}$. However, by induction hypothesis, we have $\mathbf{v}' = (v_1, \dots, v_{k-1}) \in \mathcal{H}_{k-1,n,d,s(d+1)}$ where each $v_i \in \mathcal{H}_{1,n,d,s(d+1)^{k-1}}$. Therefore, $\mathbf{v} = (\mathbf{v}', \mathbf{v}'') \in \mathcal{H}_{k,n,d,s}$ and $\mathcal{H}_{k,n,d,s}$ is a hitting set for the class of polynomials $S_{k,n,d,s}$.

Finally, note that $|\mathcal{H}_{k,n,d,s}| = |\mathcal{H}_{1,n,d,s_k}|^k$. Since $|\mathcal{H}_{1,n,d,s_k}| \leq (nds_k)^{O(\log d)}$, it follows that $|\mathcal{H}_{k,n,d,s}| \leq (nsk d)^{O(k^2 \log d)}$, and the hitting set $\mathcal{H}_{k,n,d,s}$ can be constructed in the claimed running time¹⁰. For zero testing, we need to evaluate the input ABP on the matrices of the hitting set, and this can be done in time polynomial in the input size and the size of the hitting set by matrix additions and multiplications. ◀

5 Randomized Zero Testing of Weighted Automata Over k -Monoids

We now consider pc monoids more general than k -clique monoids. A k -monoid is a pc monoid M whose non-commutation graph G_M has a 2-covering $\{G_1, G_2\}$ such that G_1 has an edge clique cover of size k' and G_2 has a vertex cover of size $k - k'$, for some k' . It follows that G_M has a k -covering of cliques and star graphs. We assume that this k -covering of G_M is given as part of the input. Let $\mathbb{F}\langle M \rangle$ denote the \mathbb{F} -algebra generated by the monoid M .

► **Lemma 5.1.** *Let $\{M_i\}_{i=1}^k$ be pc monoids defined over disjoint variable sets $\{X_i\}_{i=1}^k$, respectively. For each i , suppose \mathcal{A}_i is a randomized procedure that outputs an evaluation $v_i : \mathbb{F}\langle M_i \rangle \rightarrow \mathcal{M}_{t_i(d)}(\mathbb{F})$ such that for any polynomial g_i in $\mathbb{F}\langle M_i \rangle$ of degree at most d , g_i is nonzero if and only if $v_i(g_i)$ is a nonzero matrix with probability at least $1 - \frac{1}{2k}$.*

Then, for the evaluation $\mathbf{v} : \mathbb{F}\langle M_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle M_k \rangle \rightarrow \mathcal{M}_{t_1(d)}(\mathbb{F}) \otimes \cdots \otimes \mathcal{M}_{t_k(d)}(\mathbb{F})$ such that $\mathbf{v} = (v_1, \dots, v_k)$ and any nonzero polynomial $f \in \mathbb{F}\langle M_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle M_k \rangle$ of degree at most d , the matrix $\mathbf{v}(f)$ is nonzero with probability at least $1/2$.

¹⁰Independent to the context of the current paper, bootstrapping hitting sets has found other interesting applications in arithmetic circuit complexity [1].

Proof. The proof is by induction on k . For the base case $k = 1$, it is trivial. Let us fix an $f \in \mathbb{F}\langle M_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle M_k \rangle$ of degree at most d such that $f \neq 0$. The polynomial f can be written as $f = \sum_{m \in M_k} f_m \otimes m$ where m are the words over the pc monoid M_k and $f_m \in \mathbb{F}\langle M_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle M_{k-1} \rangle$. Since $f \neq 0$ we must have $f_m \neq 0$ for some $m \in M_k$.

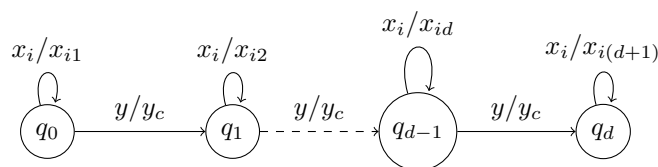
Now, inductively we have the evaluation $\mathbf{v}' = (v_1, \dots, v_{k-1})$ for the class of polynomials in $\mathbb{F}\langle M_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle M_{k-1} \rangle$ of degree at most d . Since $f_m \neq 0$, with high probability $\mathbf{v}'(f_m)$ is a nonzero matrix of dimension $\prod_{i=1}^{k-1} t_i(d)$. By induction the failure probability is bounded by $\frac{k-1}{2k}$.

As \mathbf{v}' is a *partial evaluation* for f , we observe that $\mathbf{v}'(f)$ is a matrix of dimension $\prod_{i=1}^{k-1} t_i(d)$ whose entries are polynomials in $\mathbb{F}\langle M_k \rangle$. Since $\mathbf{v}'(f_m) \neq 0$ we conclude that some $(p, q)^{th}$ entry of $\mathbf{v}'(f)$ contains a nonzero polynomial $g \in \mathbb{F}\langle M_k \rangle$ of degree at most d . Choose the evaluation $v_k \in S_k$ which is the output of the randomized procedure \mathcal{A}_k , such that $v_k(g)$ is a nonzero matrix of dimension $t_k(d)$. Hence, for the combined evaluation $\mathbf{v} = (\mathbf{v}', v_k)$, $\mathbf{v}(f)$ is a nonzero matrix of dimension $\prod_{i=1}^k t_i(d)$. A union bound shows that the failure probability is at most $1/2$. ◀

For the proof of Theorem 1.4, we first give a randomized polynomial-time identity testing algorithm for polynomials over pc monoids whose non-commutation graph is a star graph.

▶ **Lemma 5.2.** *Let $M = ((X \cup y)^*, I)$ be a monoid whose non-commutation graph G_M is a star graph with center y . Then for any constant k , there is a randomized procedure that outputs an evaluation $\mathbf{v} : X \cup \{y\} \rightarrow \text{Mat}_{t(d)}(\mathbb{F})$ where $t(d)$ is at most d , such that for any polynomial $f \in \mathbb{F}\langle M \rangle$ of degree at most d , the polynomial f is nonzero if and only if $\mathbf{v}(f)$ is a nonzero matrix. The success probability of the algorithm is at least $1 - \frac{1}{2k}$.*

Proof. If f is nonzero, then there exists a monomial m in M with nonzero coefficient. The idea is to isolate all monomials in $\{X \cup y\}^*$ that are equivalent to m in M . Let the degree of y in monomial m be $\ell \leq d$. Then m can be written as $m = m_1 y m_2 \cdots m_\ell y m_{\ell+1}$ where each m_i is a word in X^* . As X is a commuting set of variables, any permutation of m_i produces a monomial equivalent to m in M . Now consider the automaton in Figure 1.



■ **Figure 1** The transition diagram of the automaton.

Let m as $m = m_1 y m_2 \cdots m_\ell y m_{\ell+1}$, where each m_i is a maximal substring of m in X^* . We refer to the m_i as blocks. The above automaton keeps count of blocks as it scans the monomial m . As it scans m , if the automaton is in the j^{th} block, it substitutes each variable $x_i \in X$ read by a corresponding commuting variable x_{ij} where the index j encodes the block number. The y variable is renamed by a commutative variable y_c . The transition matrices N_{x_i} and N_y of dimension $d + 1$. The transition matrices are explicitly given below.

$$N_{x_i} = \begin{bmatrix} x_{i1} & 0 & 0 & \dots & 0 \\ 0 & x_{i2} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & x_{id} & 0 \\ 0 & 0 & \dots & 0 & x_{i(d+1)} \end{bmatrix}, \quad N_y = \begin{bmatrix} 0 & y_c & 0 & \dots & 0 \\ 0 & 0 & y_c & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & y_c \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}.$$

10:12 Equivalence Testing of Weighted Automata

Now we explain this matrix substitution. Let $f = \sum_m \alpha_m m$, where $\alpha_m \in \mathbb{F}$. We write $f = \sum_{\ell=1}^d f_\ell$, where $f_\ell = \sum_{m: \deg_y(m)=\ell} \alpha_m m$. That is, f_ℓ is the part of f consisting of monomials m with y -degree $\deg_y(m) = \ell$.

From the description of the automaton, we can see that for each $\ell \in [d]$, the $(0, \ell)^{th}$ entry of the output matrix is the commutative polynomial $f_\ell^c \in \mathbb{F}[\{x_{i,j}\}_{1 \leq i \leq n, 1 \leq j \leq d+1}, y_c]$. The construction ensures the following: For each $0 \leq \ell \leq d$, $f_\ell = 0$ if and only if $f_\ell^c = 0$.

The randomized identity test is by substituting random scalar values for the commuting variables x_{ij} and y_c from a set $S \subseteq \mathbb{F}$ of size at least $2kd$, such that the output matrix becomes nonzero. The bound on the success probability follows from Polynomial Identity Lemma [25, 20, 7]. ◀

Now we are ready to prove Theorem 1.4.

Proof of Theorem 1.4. Let M' be a pc monoid whose non-commutation graph $G_{M'}$ is a clique. Let $g \in \mathbb{F}\langle M' \rangle$ be a nonzero polynomial of degree at most d . By the Amitsur-Levitzki Theorem [2], if we substitute variables $x_i \in M'$ by generic matrix of size d over the variables $\{x_{u,v}^{(i)}\}_{1 \leq u,v \leq d}$, the output matrix is nonzero¹¹. Moreover, the entries of the output matrix are commutative polynomials of degree at most d in the variables $\{x_{u,v}^{(i)}\}_{1 \leq i \leq n, 1 \leq u,v \leq d}$. It suffices to randomly substitute for each $x_{u,v}^{(i)}$ variable from a set $S \subseteq \mathbb{F}$ of size at least $2kd$. This defines the evaluation map $v : \mathbb{F}\langle M' \rangle \rightarrow \mathbb{M}_d(\mathbb{F})$. The resulting identity test succeeds with probability at least $1 - \frac{1}{2k}$. For the star graphs, the evaluation map is already defined in Lemma 5.2.

Given a \mathbb{F} -weighted automaton A of size s over a k -monoid $M = (X^*, I)$, by Theorem 3.5, the zero testing of A reduces to identity testing of a collection of ABPs of the form : $f = \lambda^T N^d \mu$ over $\mathbb{F}\langle M \rangle$, where N is the transition matrix of A and d is bounded by $O(ns^2k)$. Now, to test identity of f where M is a k -monoid, it suffices to test identity of $\psi(f)$ where ψ is the injective homomorphism from Lemma 3.1. Now $\psi(f)$ in $\mathbb{F}\langle M'_1 \rangle \otimes \cdots \otimes \mathbb{F}\langle M'_k \rangle$, where for each $i \in [k]$ the non-commutation graph of M'_i is either a clique or a star.

By Lemma 5.1, we can construct the evaluation map $\mathbf{v} = v_1 \otimes v_2 \otimes \cdots \otimes v_k$ where for each $i \in [k]$, v_i is an evaluation map for either a clique or a star graph depending on M'_i . The range of \mathbf{v} is matrices of dimension at most d^k , which is bounded by $(sn)^{O(k)}$ as d is bounded by $O(ns^2k)$. This completes the proof of Theorem 1.4. ◀

Concluding Remarks

The bootstrapped construction presented in Section 4 designs a quasi-polynomial time algorithm for the k -clique monoid problem which uses evaluation over a suitable matrix algebra. However, to design a polynomial-time algorithm, one may try to exploit finer structures in the problem than evaluating it over a matrix algebra. The obvious natural idea is to generalize the white-box polynomial-time identity testing algorithm for noncommutative ABPs [18] in the pc monoid setting. However, it is unclear whether such a generalization is possible.

¹¹ In fact the Amitsur-Levitzki theorem guarantees that generic matrices of size $\lceil \frac{d}{2} \rceil + 1$ suffice [2].

References

- 1 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM J. Comput.*, 44(3):669–697, 2015. doi:10.1137/140975103.
- 2 A. S. Amitsur and J. Levitzki. Minimal identities for algebras. *Proceedings of the American Mathematical Society*, 1(4):449–463, 1950. URL: <http://www.jstor.org/stable/2032312>.
- 3 C. Beeri. An improvement on Valiant’s decision procedure for equivalence of deterministic finite turn pushdown machines. *Theoretical Computer Science*, 3(3):305–320, 1976. doi:10.1016/0304-3975(76)90049-9.
- 4 J. Berstel and C. Reutenauer. *Noncommutative Rational Series with Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011.
- 5 Malcolm Bird. The equivalence problem for deterministic two-tape automata. *J. Comput. Syst. Sci.*, 7(2):218–236, 1973. doi:10.1016/S0022-0000(73)80045-5.
- 6 Mireille Clerbout, Michel Latteux, and Yves Roos. Decomposition of partial commutations. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 501–511. Springer, 1990. doi:10.1007/BFb0032054.
- 7 Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.
- 8 Volker Diekert. *Combinatorics on Traces*, volume 454 of *Lecture Notes in Computer Science*. Springer, 1990. doi:10.1007/3-540-53031-2.
- 9 Volker Diekert, Markus Lohrey, and Alexander Miller. Partially commutative inverse monoids. In Rastislav Kralovic and Pawel Urzyczyn, editors, *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2006. doi:10.1007/11821069_26.
- 10 Manfred Droste and Paul Gastin. On recognizable and rational formal power series in partially commuting variables. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP’97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 682–692. Springer, 1997. doi:10.1007/3-540-63165-8_222.
- 11 Samuel Eilenberg. *Automata, Languages, and Machines (Vol A)*. Pure and Applied Mathematics. Academic Press, 1974.
- 12 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013. doi:10.1109/FOCS.2013.34.
- 13 Emily P. Friedman and Sheila A. Greibach. A polynomial time algorithm for deciding the equivalence problem for 2-tape deterministic finite state acceptors. *SIAM J. Comput.*, 11:166–183, 1982.
- 14 T. V. Griffiths. The unsolvability of the equivalence problem for nondeterministic generalized machines. *J. ACM*, 15(3):409–413, 1968. doi:10.1145/321466.321473.
- 15 Tero Harju and Juhani Karhumäki. The equivalence problem of multitape finite automata. *Theor. Comput. Sci.*, 78(2):347–355, 1991. doi:10.1016/0304-3975(91)90356-7.
- 16 Antoni W. Mazurkiewicz. Trace theory. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986*, pages 279–324, 1986. doi:10.1007/3-540-17906-2_30.
- 17 Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418, 1991. doi:10.1145/103418.103462.
- 18 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. doi:10.1007/s00037-005-0188-8.

- 19 M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245–270, 1961. doi:10.1016/S0019-9958(61)80020-X.
- 20 Jacob T. Schwartz. Fast probabilistic algorithm for verification of polynomial identities. *J. ACM.*, 27(4):701–717, 1980.
- 21 W. Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.
- 22 Leslie G. Valiant. The equivalence problem for deterministic finite-turn pushdown automata. *Information and Control*, 25(2):123–133, 1974. doi:10.1016/S0019-9958(74)90839-0.
- 23 Stefano Varricchio. On the decidability of equivalence problem for partially commutative rational power series. *Theor. Comput. Sci.*, 99(2):291–299, 1992. doi:10.1016/0304-3975(92)90354-I.
- 24 James Worrell. Revisiting the equivalence problem for finite multitape automata. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 422–433, 2013. doi:10.1007/978-3-642-39212-2_38.
- 25 R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. of the Int. Sym. on Symbolic and Algebraic Computation*, pages 216–226, 1979.

A The Proof of Lemma 3.1

Proof. It is straightforward to check that ψ is a ring homomorphism. To show the injectivity, it is enough to show that $\psi(m) = \psi(m')$ implies $m = m'$ in M for any words $m, m' \in M$. We prove the claim by induction on the length of words in M . Suppose that for words $m \in M$ of length at most ℓ , if m' is not \tilde{I} -equivalent to m then $\psi(m) \neq \psi(m')$. The base case, for $\ell = 0$ clearly holds.

Now, suppose $m = x \cdot m_1 \in X^{\ell+1}$ for $x \in X$ and $\psi(m) = \psi(m')$.

▷ **Claim A.1.** For some $m_2 \in M$, $m' = x \cdot m_2$ in M .

Proof. Assume, to the contrary, that there is no $m_2 \in M$ such that $m' = x \cdot m_2$. Let $J = \{j \in [k] \mid x \in X_j\}$. If the variable x does not occur in m' then $m|_{X_j} \neq m'|_{X_j}$ for each $j \in J$. This implies that $\psi(m) \neq \psi(m')$ which is a contradiction.

On other hand, suppose x occurs in m' and it cannot be moved to the leftmost position in m' applying the commutation relations in I . Then we must have $m' = ayxb$ for some $y \in X_j$ and $j \in J$, where $a, b \in X^*$, for the leftmost occurrence of x in m' . Hence $m|_{X_j} \neq m'|_{X_j}$, because x is the first variable in $m|_{X_j}$ and x comes after y in $m'|_{X_j}$. Therefore, $\psi(m) \neq \psi(m')$ which is a contradiction. ◁

Now, $\psi(x \cdot m_1) = \psi(x \cdot m_2)$ implies that $\psi(m_1) = \psi(m_2)$. Both m_1 and m_2 are of length ℓ . By induction hypothesis it follows that $m_1 = m_2$, and hence $m = m'$. ◀

B The Proof of Lemma 4.3

Proof. In effect the edges of the input branching program B are now labelled by matrices of dimension T with entries are linear forms over the variables X'_k . To show that each entry of the final $T \times T$ matrix can be computed by an ABP of size sT , let us fix some (i, j) such that $1 \leq i, j \leq T$ and construct an ABP B'_{ij} computing the polynomial in the $(i, j)^{th}$ entry.

The construction of B'_{ij} is as follows. We make T copies of each node p (except the source and sink node) of B and label it as (p, k) for each $k \in [T]$. Let us fix two nodes p and q from B such that there is a $T \times T$ matrix M_{pq} labelling the edge (p, q) after the substitution. Then, for each $j_1, j_2 \in [T]$, add an edge between (p, j_1) and (q, j_2) in B'_{ij} and label it by the $(j_1, j_2)^{th}$ entry of M_{pq} . When p is the source node, for each $j_2 \in T$, add an edge between the

source node and (q, j_2) in B'_{ij} and label it by the $(i, j_2)^{th}$ entry of M_{pq} . Similarly, when q is the sink node, for each $j_1 \in T$, add an edge between (p, j_1) and the sink node in B'_{ij} and label it by the $(j_1, j)^{th}$ entry of M_{pq} .

We just need to argue that the intermediate edge connections simulate matrix multiplications correctly. This is simple to observe, since for each path

$$\mathcal{P} = \{(s, p_1), (p_1, p_2), \dots, (p_{\ell-1}, t)\}$$

in B (where s, t are the source and sink nodes respectively) and each $(j_1, \dots, j_{\ell-1})$ such that $1 \leq j_1, \dots, j_{\ell-1} \leq T$, there is a path $(s, (p_1, j_1)), ((p_1, j_1), (p_2, j_2)), \dots, ((p_{\ell-1}, j_{\ell-1}), t)$ in B'_{ij} that computes $M_{(s, p_1)}[i, j_1]M_{(p_1, p_2)}[j_1, j_2] \cdots M_{(p_{\ell-1}, t)}[j_{\ell-1}, j]$ where $M_{(p, q)}$ is the $T \times T$ matrix labelling the edge (p, q) in B . The size of B'_{ij} is sT , and the number of layers is $d + 1$. ◀

Finitely Tractable Promise Constraint Satisfaction Problems

Kristina Asimi ✉

Department of Algebra, Faculty of Mathematics and Physics, Charles University, Prague, Czechia

Libor Barto ✉ 🏠 

Department of Algebra, Faculty of Mathematics and Physics, Charles University, Prague, Czechia

Abstract

The Promise Constraint Satisfaction Problem (PCSP) is a generalization of the Constraint Satisfaction Problem (CSP) that includes approximation variants of satisfiability and graph coloring problems. Barto [LICS '19] has shown that a specific PCSP, the problem to find a valid Not-All-Equal solution to a 1-in-3-SAT instance, is not finitely tractable in that it can be solved by a trivial reduction to a tractable CSP, but such a CSP is necessarily over an infinite domain (unless $P=NP$). We initiate a systematic study of this phenomenon by giving a general necessary condition for finite tractability and characterizing finite tractability within a class of templates – the “basic” tractable cases in the dichotomy theorem for symmetric Boolean PCSPs allowing negations by Brakensiek and Guruswami [SODA'18].

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Constraint and logic programming

Keywords and phrases Constraint satisfaction problems, promise constraint satisfaction, Boolean PCSP, polymorphism, finite tractability, homomorphic relaxation

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.11

Funding Both authors have received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 771005).



1 Introduction

Many computational problems, including various versions of logical satisfiability, graph coloring, and systems of equations can be phrased as Constraint Satisfaction Problems (CSPs) over fixed templates (see [5]). One of the possible formulations of the CSP is via homomorphisms of relational structures: a *template* \mathbb{A} is a relational structure with finitely many relations and the CSP over \mathbb{A} , written $\text{CSP}(\mathbb{A})$, is the problem to decide whether a given finite relational structure \mathbb{X} (similar to \mathbb{A}) admits a homomorphism to \mathbb{A} .

The complexity of CSPs over finite templates (i.e., those templates whose domain is a finite set) is now completely classified by a celebrated dichotomy theorem independently obtained by Bulatov [10] and Zhuk [19, 20]: every $\text{CSP}(\mathbb{A})$ is either tractable (that is, solvable in polynomial-time) or NP-complete. The landmark results leading to the complete classification include Schaefer’s dichotomy theorem [18] for CSPs over Boolean structures (i.e., structures with a two-element domain), Hell and Nešetřil’s dichotomy theorem [15] for CSPs over graphs, and Feder and Vardi’s thorough study [13] through Datalog and group theory. The latter paper also inspired the development of a mathematical theory of finite-template CSPs [16, 9, 6], the so called *algebraic approach*, that provided guidance and tools for the general dichotomy theorem by Bulatov and Zhuk.

The algebraic approach has been successfully applied in many variants and generalizations of the CSP such as the infinite-template CSP [7] or valued CSP [17]. This paper concerns a recent vast generalization of the basic CSP framework, the Promise CSP (PCSP).



© Kristina Asimi and Libor Barto;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 11; pp. 11:1–11:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A *template* for the PCSP is a pair (\mathbb{A}, \mathbb{B}) of similar structures such that \mathbb{A} has a homomorphism to \mathbb{B} , and the PCSP over (\mathbb{A}, \mathbb{B}) , written $\text{PCSP}(\mathbb{A}, \mathbb{B})$, is the problem to distinguish between the case that a given finite structure \mathbb{X} admits a homomorphism to \mathbb{A} and the case that \mathbb{X} does not have a homomorphism to \mathbb{B} (the promise is that one of the cases takes place). This framework generalizes that of CSP (take $\mathbb{A} = \mathbb{B}$) and additionally includes important problems in approximation, e.g., if $\mathbb{A} = \mathbb{K}_k$ (the clique on k vertices) and $\mathbb{B} = \mathbb{K}_l$, $k \leq l$, then $\text{PCSP}(\mathbb{A}, \mathbb{B})$ is a version of the approximate graph coloring problem, namely, the problem to distinguish graphs that are k -colorable from those that are not l -colorable, a problem whose complexity is open after more than 40 years of research. On the other hand, the basics of the algebraic approach to CSPs can be generalized to PCSPs [1, 8, 11, 3].

The approximate graph coloring problem shows that a full classification of the complexity of PCSPs over graph templates is still open and so is the analogue of Schaefer’s Boolean CSP, PCSPs over pairs of Boolean structures. However, strong partial results have already been obtained. Brakensiek and Guruswami [8] proved a dichotomy theorem for all symmetric Boolean templates allowing negations, i.e., templates (\mathbb{A}, \mathbb{B}) such that $\mathbb{A} = (\{0, 1\}; R_0, R_1, \dots)$, $\mathbb{B} = (\{0, 1\}; S_0, S_1, \dots)$, each relation R_i, S_i is invariant under permutations of coordinates, and $R_0 = S_0$ is the binary disequality relation \neq . Fıcak, Kozik, Olřák, and Stankiewicz [14] later generalized this result to all symmetric Boolean templates. These templates play a central role in this paper.

To prove tractability or hardness results for PCSPs, a very simple but useful reduction is often applied: If (\mathbb{A}, \mathbb{B}) and $(\mathbb{A}', \mathbb{B}')$ are similar PCSP templates and there exist homomorphisms $\mathbb{A}' \rightarrow \mathbb{A}$ and $\mathbb{B} \rightarrow \mathbb{B}'$, then the trivial reduction (which does not change the instance) reduces $\text{PCSP}(\mathbb{A}', \mathbb{B}')$ to $\text{PCSP}(\mathbb{A}, \mathbb{B})$; we say that $(\mathbb{A}', \mathbb{B}')$ is a *homomorphic relaxation* of (\mathbb{A}, \mathbb{B}) . In fact, all the tractable symmetric Boolean PCSPs can be reduced in this way to a tractable CSP over a structure with a *possibly infinite domain*.

An interesting example of a PCSP that can be naturally reduced to a tractable CSP over an infinite domain is the following problem. An instance is a list of triples of variables and the problem is to distinguish instances that are satisfiable as positive 1-in-3-SAT instances from those that are not even satisfiable as Not-All-Equal-3-SAT instances. This computational problem is essentially the same as $\text{PCSP}(\mathbb{A}, \mathbb{B})$ where \mathbb{A} consists of the ternary 1-in-3 relation over $\{0, 1\}$ and \mathbb{B} consists of the ternary not-all-equal relation over $\{0, 1\}$. It is easy to see that $\mathbb{A} \rightarrow \mathbb{C} \rightarrow \mathbb{B}$ where \mathbb{C} is the relation “ $x + y + z = 1$ ” over the set of all integers. Therefore $\text{PCSP}(\mathbb{A}, \mathbb{B})$ is reducible (by means of the trivial reduction) to $\text{PCSP}(\mathbb{C}, \mathbb{C}) = \text{CSP}(\mathbb{C})$ which is a tractable problem. The main result of [2] is that no finite structure can be used in place of \mathbb{C} for this particular template – this PCSP is not finitely tractable in the sense of the following definition.

► **Definition 1.** *We say that $\text{PCSP}(\mathbb{A}, \mathbb{B})$ is finitely tractable if there exists a finite relational structure \mathbb{C} such that $\mathbb{A} \rightarrow \mathbb{C} \rightarrow \mathbb{B}$ and $\text{CSP}(\mathbb{C})$ is tractable. Otherwise we call $\text{PCSP}(\mathbb{A}, \mathbb{B})$ not finitely tractable. (We assume $P \neq NP$ throughout the paper.)*

In this paper, we initiate a systematic study of this phenomenon. As the main technical contribution, we determine which of the “basic tractable cases” in Brakensiek and Guruswami’s classification [8] are finitely tractable. It turns out that finite tractability is quite rare, so the infinite nature of the 1-in-3 versus Not-All-Equal problem is not exceptional at all.

1.1 Symmetric Boolean PCSPs allowing negations

We now discuss the classification of symmetric Boolean templates allowing negations from [8]. It will be convenient to describe these templates by listing the corresponding relation pairs, that is, instead of $(\mathbb{A} = (\{0, 1\}; R_1, \dots, R_n), \mathbb{B} = (\{0, 1\}; S_1, \dots, S_n))$ we describe this

template by the list $(R_1, S_1), \dots, (R_n, S_n)$. Recall that the template is *symmetric* if all the involved relations are symmetric, i.e., invariant under any permutation of coordinates, and the template *allows negations* if (\neq, \neq) is among the relation pairs, where $\neq = \{(0, 1), (1, 0)\}$ is the disequality relation.

It may be also helpful to think of an instance of $\text{PCSP}(\mathbb{A}, \mathbb{B})$ as a list of constraints of the form $R_i(\text{variables})$ and the problem is to distinguish between instances where each constraint is satisfiable and those which are not satisfiable even when we replace each R_i by the corresponding “relaxed version” S_i . Allowing negations then means that we can use constraints $x \neq y$ – we can effectively negate variables.

The following relations are important for the classification.

- odd-in- $s = \{\mathbf{x} \in \{0, 1\}^s : \sum_{i=1}^s x_i \text{ is odd}\}$, even-in- $s = \{\mathbf{x} \in \{0, 1\}^s : \sum_{i=1}^s x_i \text{ is even}\}$
- r -in- $s = \{\mathbf{x} \in \{0, 1\}^s : \sum_{i=1}^s x_i = r\}$
- $\leq r$ -in- $s = \{\mathbf{x} \in \{0, 1\}^s : \sum_{i=1}^s x_i \leq r\}$, $\geq r$ -in- $s = \{\mathbf{x} \in \{0, 1\}^s : \sum_{i=1}^s x_i \geq r\}$
- not-all-equal- $s = \{\mathbf{x} \in \{0, 1\}^s : \sum_{i=1}^s x_i \notin \{0, s\}\}$

The next theorem lists some of the tractable cases of the classification, which are “basic” in the sense explained below.

► **Theorem 2** ([8]). $\text{PCSP}((P, Q), (\neq, \neq))$ is tractable if (P, Q) is equal to

- (a) (odd-in- s , odd-in- s), or (even-in- s , even-in- s), or
- (b) ($\leq r$ -in- s , $\leq (2r - 1)$ -in- s) and $r \leq s/2$, or
($\geq r$ -in- s , $\geq (2r - s + 1)$ -in- s) and $r \geq s/2$, or
- (c) (r -in- s , not-all-equal- s)

for some positive integers r, s .

It follows from the results in [8] (namely Theorem 2.1 and a simple analysis of compatible relations) that every tractable symmetric Boolean PCSP allowing negations can be obtained by

- taking any number of (\neq, \neq) and any number of relation pairs from a fixed item in Theorem 2,
- adding any number of “trivial” relation pairs (P, Q) such that $P \subseteq Q$, and Q is the full relation or P contains only constant tuples, and
- taking a homomorphic relaxation of the obtained template.

In this sense, Theorem 2 provides building blocks for all tractable templates.

1.2 Contributions

Some of the cases in Theorem 2 are finitely tractable: templates in item (a) are tractable CSPs (they can be decided by solving systems of linear equations of the two-element field), templates in item (c) for r odd and s even are homomorphic relaxations of (odd-in- s , odd-in- s), and templates in item (b) for $r = 1$ or $r = s - 1$ as well as all templates with $s \leq 2$ are tractable CSPs (reducible to 2-SAT) [18, 5]. Our main theorem proves that all the remaining cases are not finitely tractable. In fact, we prove this property even for some relaxations of these templates:

► **Theorem 3.** *The PCSP over any of the following templates is not finitely tractable.*

- (1) (r -in- s , $\leq (2r - 1)$ -in- s), (\neq, \neq) where $1 < r < s/2$,
(r -in- s , $\geq (2r - s + 1)$ -in- s), (\neq, \neq) where $s/2 < r < s - 1$
- (2) ($\leq r$ -in- s , $\leq (2r - 1)$ -in- s), (\neq, \neq) where s is even, $1 < r = s/2$
($\geq r$ -in- s , $\geq (2r - s + 1)$ -in- s), (\neq, \neq) where s is even, $1 < r = s/2$

- (3) (r -in- s , $\leq(2r - 1)$ -in- s), (\neq, \neq) where s is even, $1 < r = s/2$, and r is even
 $(r$ -in- s , $\geq(2r - s + 1)$ -in- s), (\neq, \neq) where s is even, $1 < r = s/2$, and r is even
- (4) (r -in- s , not-all-equal- s) where $s > r$, $s > 2$, and r is even or s is odd

Note that the templates in the last item do not contain the disequality pair; the special case with $r = 1$ and $s = 3$ is the main result of [2]. Disequalities in the other items are necessary, since otherwise the templates are homomorphic relaxations of CSPs over one-element structures.

In Theorem 18 we provide a general necessary condition for finite tractability of an arbitrary finite-template PCSP in terms of so called h1 identities. Showing that templates in Theorem 3 do not satisfy this necessary condition forms the bulk of the paper.

The necessary condition in Theorem 18 seems very unlikely to be sufficient for finite tractability. Nevertheless, we observe in Theorem 12 that finite tractability *does* depend only on h1 identities, just like standard tractability [11], see Theorem 10 and the discussion following the theorem.

2 Preliminaries

2.1 PCSP

We use the notation $[n] = \{1, 2, \dots, n\}$ throughout the paper.

A *relational structure* (of finite signature) is a tuple $\mathbb{A} = (A; R_1, R_2, \dots, R_n)$ where A is a set, called the *domain*, and each R_i is a relation on A of arity $\text{ar}(R_i) \geq 1$, that is, $R_i \subseteq A^{\text{ar}(R_i)}$. The structure \mathbb{A} is finite if A is finite. Two relational structures $\mathbb{A} = (A; R_1, R_2, \dots, R_n)$ and $\mathbb{B} = (B; S_1, S_2, \dots, S_n)$ are *similar* if they have the same number of relations and $\text{ar}(R_i) = \text{ar}(S_i)$ for each $i \in [n]$. In this case, a *homomorphism* from \mathbb{A} to \mathbb{B} is a mapping $f : A \rightarrow B$ such that $(f(a_1), f(a_2), \dots, f(a_k)) \in S_i$ whenever $i \in [n]$ and $(a_1, a_2, \dots, a_k) \in R_i$ where $k = \text{ar}(R_i)$. If there exists a homomorphism from \mathbb{A} to \mathbb{B} , we write $\mathbb{A} \rightarrow \mathbb{B}$, and if there is none, we write $\mathbb{A} \not\rightarrow \mathbb{B}$.

► **Definition 4.** A PCSP template is a pair (\mathbb{A}, \mathbb{B}) of similar relational structures such that $\mathbb{A} \rightarrow \mathbb{B}$.

The PCSP over (\mathbb{A}, \mathbb{B}) , written $\text{PCSP}(\mathbb{A}, \mathbb{B})$, is the following problem. Given a finite relational structure \mathbb{X} similar to \mathbb{A} (and \mathbb{B}), output “Yes.” if $\mathbb{X} \rightarrow \mathbb{A}$ and output “No.” if $\mathbb{X} \not\rightarrow \mathbb{B}$.

We define $\text{CSP}(\mathbb{A}) = \text{PCSP}(\mathbb{A}, \mathbb{A})$.

► **Definition 5.** Let (\mathbb{A}, \mathbb{B}) and $(\mathbb{A}', \mathbb{B}')$ be similar PCSP templates. We say that $(\mathbb{A}', \mathbb{B}')$ is a homomorphic relaxation of (\mathbb{A}, \mathbb{B}) if $\mathbb{A}' \rightarrow \mathbb{A}$ and $\mathbb{B} \rightarrow \mathbb{B}'$.

Recall that if $(\mathbb{A}', \mathbb{B}')$ is a homomorphic relaxation of (\mathbb{A}, \mathbb{B}) , then the trivial reduction, which does not change the input structure \mathbb{X} , reduces $\text{PCSP}(\mathbb{A}', \mathbb{B}')$ to $\text{PCSP}(\mathbb{A}, \mathbb{B})$.

2.2 Polymorphisms

A crucial concept for the algebraic approach to (P)CSP is a polymorphism.

► **Definition 6.** Let $\mathbb{A} = (A; R_1, \dots, R_m)$ and $\mathbb{B} = (B; S_1, \dots, S_m)$ be two similar relational structures. A function $c : A^n \rightarrow B$ is a polymorphism from \mathbb{A} to \mathbb{B} if for each relation R_i in \mathbb{A} with $k_i = \text{arity}(R_i)$

$$\begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{k_i1} \end{pmatrix} \in R_i, \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{k_i2} \end{pmatrix} \in R_i \dots, \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{k_in} \end{pmatrix} \in R_i \Rightarrow \begin{pmatrix} c(a_{11}, a_{12}, \dots, a_{1n}) \\ c(a_{21}, a_{22}, \dots, a_{2n}) \\ \vdots \\ c(a_{k_i1}, a_{k_i2}, \dots, a_{k_in}) \end{pmatrix} \in S_i.$$

We denote the set of all polymorphisms from \mathbb{A} to \mathbb{B} by $\text{Pol}(\mathbb{A}, \mathbb{B})$ and define $\text{Pol}(\mathbb{C}) = \text{Pol}(\mathbb{C}, \mathbb{C})$.

The computational complexity of a PCSP depends only on the set of polymorphisms of its template [8]. We note that tractability of the PCSPs in Theorem 2 stems from nice polymorphisms: parities (item (a)), majorities (item (b)), and alternating thresholds (item (c)).

The set of polymorphisms is an algebraic object named minion in [11], which we define in Definition 8 below.

► **Definition 7.** An n -ary function $f^\pi : A^n \rightarrow B$ is called a minor of an m -ary function $f : A^m \rightarrow B$ given by a map $\pi : [m] \rightarrow [n]$ if

$$f^\pi(x_1, \dots, x_n) = f(x_{\pi(1)}, \dots, x_{\pi(m)})$$

for all $x_1, \dots, x_n \in A$.

► **Definition 8.** Let $\mathcal{O}(A, B) = \{f : A^n \rightarrow B : n \geq 1\}$. A minion on (A, B) is a non-empty subset \mathcal{M} of $\mathcal{O}(A, B)$ that is closed under taking minors. For fixed $n \geq 1$, let $\mathcal{M}^{(n)}$ denote the set of n -ary functions from \mathcal{M} .

As mentioned, $\mathcal{M} = \text{Pol}(\mathbb{A}, \mathbb{B})$ is always a minion and the complexity of $\text{PCSP}(\mathbb{A}, \mathbb{B})$ depends only on \mathcal{M} . This result was strengthened in [11, 3] (generalizing the same result for CSPs [6]) as follows.

► **Definition 9.** Let \mathcal{M} and \mathcal{N} be two minions. A mapping $\xi : \mathcal{M} \rightarrow \mathcal{N}$ is called a minion homomorphism if it preserves arities and preserves taking minors, i.e., $\xi(f^\pi) = (\xi(f))^\pi$ for every $f \in \mathcal{M}^{(m)}$ and every $\pi : [m] \rightarrow [n]$.

► **Theorem 10.** Let (\mathbb{A}, \mathbb{B}) and $(\mathbb{A}', \mathbb{B}')$ be PCSP templates. If there exists a minion homomorphism $\text{Pol}(\mathbb{A}', \mathbb{B}') \rightarrow \text{Pol}(\mathbb{A}, \mathbb{B})$, then $\text{PCSP}(\mathbb{A}, \mathbb{B})$ is log-space reducible to $\text{PCSP}(\mathbb{A}', \mathbb{B}')$.

An *h1 identity* (h1 stands for height one) is a meaningful expression of the form $\text{function}(\text{variables}) \approx \text{function}(\text{variables})$, e.g., if $f : A^3 \rightarrow B$ and $g : A^4 \rightarrow B$, then $f(x, y, x) \approx g(y, x, x, z)$ is an h1 identity. Such an h1 identity is *satisfied* if the corresponding equation holds universally, e.g., $f(x, y, x) \approx g(y, x, x, z)$ is satisfied if and only if $f(x, y, x) = g(y, x, x, z)$ for every $x, y, z \in A$.

Every minion homomorphism $\xi : \mathcal{M} \rightarrow \mathcal{N}$ preserves h1 identities in the sense that if functions $f, g \in \mathcal{M}$ satisfy an h1 identity, then so do their ξ -images $\xi(f), \xi(g) \in \mathcal{N}$. In fact, an arity-preserving ξ between minions is a minion homomorphism if and only if it preserves h1 identities (see [6] for details). In this sense, Theorem 10 shows that the complexity of a PCSP depends only on h1 identities satisfied by polymorphisms.

2.3 Notation for tuples

Repeated entries in tuples will be indicated by \times , e.g. $(2 \times a, 3 \times b)$ stands for the tuple (a, a, b, b, b) .

The i -th *cyclic shift* of a tuple (x_1, \dots, x_m) is the tuple $(x_{(m-i \bmod m)+1}, \dots, x_m, x_1, \dots, x_{(m-i-1 \bmod m)+1})$. A *cyclic shift* is the i -th cyclic shift for some i . We will use cyclic shifts both for tuples of zeros and ones and tuples of variables.

We will often use special p -tuples and $n = p^2$ -tuples of zeros and ones as arguments for Boolean functions, where p will be a fixed prime number. For $0 \leq k \leq p$, $0 \leq l \leq p^2$, and $0 \leq k^1, \dots, k^p \leq p$ we write

$$\langle k \rangle_p = (k \times 1, (p - k) \times 0) = (\underbrace{1, 1, \dots, 1}_k, \underbrace{0, 0, \dots, 0}_{p-k}), \quad \langle l \rangle_n = (\underbrace{1, 1, \dots, 1}_l, \underbrace{0, 0, \dots, 0}_{n-l})$$

and

$$\langle k^1, \dots, k^p \rangle_p = \langle k^1 \rangle_p \langle k^2 \rangle_p \dots \langle k^p \rangle_p$$

for the concatenation of $\langle k^1 \rangle_p, \dots, \langle k^p \rangle_p$. (Note here that the “ i ” in k^i is an index, not an exponent.) The subscripts p and n in $\langle \rangle_p$ and $\langle \rangle_n$ will be usually clear from the context and we omit them. We will sometimes need to shift n -ary tuples $\langle k^1, \dots, k^p \rangle$ blockwise, e.g., to $\langle k^2, \dots, k^p, k^1 \rangle$. In such a situation we talk about a *p-ary cyclic shift* to avoid confusion.

It will be often convenient to think of an n -tuple $\mathbf{k} = \langle k^1, \dots, k^p \rangle$ as a $p \times p$ zero-one matrix with columns $\langle k^1 \rangle, \dots, \langle k^p \rangle$. For example, the ones in $\langle p \times 5 \rangle$ form a $5 \times p$ “rectangle” and $\langle (p-2) \times 5, 2 \times 4 \rangle$ is “almost” a $5 \times p$ rectangle – the bottom right 1×2 corner is removed. A p -ary cyclic shift of \mathbf{k} corresponds to cyclic permutation of columns.

The area of a zero-one n -tuple \mathbf{k} is defined as the fraction of ones and is denoted $\lambda(\mathbf{k})$.

$$\lambda(\mathbf{k}) = \left(\sum_{i=1}^n k_i \right) / p^2$$

The area of $\langle k^1, \dots, k^p \rangle$ is thus $(k^1 + \dots + k^p) / p^2$.

If t is a p -ary function we simply write $t\langle k \rangle$ instead of $t(\langle k \rangle)$. Similar shorthand is used for n -ary functions and tuples $\langle k^1, \dots, k^p \rangle_p$.

3 Finitely tractable PCSPs

3.1 Finite tractability depends only on h1 identities

We start by observing that finite tractability also depends only on h1 identities satisfied by polymorphisms, just like standard tractability (recall the discussion about h1 identities and minion homomorphisms below Theorem 10). This result, Theorem 12, is an immediate consequence of the following lemma and Theorem 10.

- **Lemma 11.** *Let (\mathbb{A}, \mathbb{B}) be a PCSP template. Then the following are equivalent.*
- PCSP (\mathbb{A}, \mathbb{B}) is finitely tractable.
 - There exists a finite relational structure \mathbb{C} such that CSP (\mathbb{C}) is solvable in polynomial time and there exists a minion homomorphism $\text{Pol}(\mathbb{C}) \rightarrow \text{Pol}(\mathbb{A}, \mathbb{B})$.

Proof. This lemma is a consequence of known results and we only sketch the argument here. In Section II.B of [2] it is argued that the first item is equivalent to the claim that a finite tractable template (\mathbb{C}, \mathbb{C}) pp-constructs (\mathbb{A}, \mathbb{B}) . The latter claim is equivalent to the second item by Theorem 4.12 in [3]. ◀

► **Theorem 12.** *Let (\mathbb{A}, \mathbb{B}) and $(\mathbb{A}', \mathbb{B}')$ be PCSP templates. If there exists a minion homomorphism $\text{Pol}(\mathbb{A}', \mathbb{B}') \rightarrow \text{Pol}(\mathbb{A}, \mathbb{B})$ and $\text{PCSP}(\mathbb{A}', \mathbb{B}')$ is finitely tractable, then so is $\text{PCSP}(\mathbb{A}, \mathbb{B})$.*

3.2 Necessary condition for finite tractability

In this subsection, we derive the necessary condition for finite tractability that will be used to prove Theorem 3. A cyclic polymorphism is a starting point for the condition.

► **Definition 13.** *A function $c : A^p \rightarrow B$ is called cyclic if it satisfies the h1 identity*

$$c(x_1, x_2, \dots, x_p) \approx c(x_2, \dots, x_p, x_1).$$

Cyclic polymorphisms can be used [4] to characterize the borderline between tractable and NP-complete CSPs proposed in [9] and confirmed in [10, 19, 20]. We only state the direction needed in this paper.

► **Theorem 14 ([4]).** *Let \mathbb{C} be a CSP template over a finite domain C . If $\text{CSP}(\mathbb{C})$ is not NP-complete, then \mathbb{C} has a cyclic polymorphism of arity p for every prime number $p > |C|$.*

Polymorphism minions of CSP templates are closed under arbitrary composition (cf. [5]). In particular, if $\text{CSP}(\mathbb{C})$ is not NP-complete, then $\text{Pol}(\mathbb{C})$ contains the function

$$\begin{aligned} t(x_{11}, x_{21}, \dots, x_{p1}, x_{12}, x_{22}, \dots, x_{p2}, \dots, x_{1p}, x_{2p}, \dots, x_{pp}) \\ = c(c(x_{11}, x_{21}, \dots, x_{p1}), c(x_{12}, x_{22}, \dots, x_{p2}), \dots, c(x_{1p}, x_{2p}, \dots, x_{pp})), \end{aligned} \quad (1)$$

where c is a p -ary cyclic function and $p > |C|$. Such a function satisfies strong h1 identities which are not satisfied by the templates in Theorem 3. We now (in two steps) describe one such collection of strong enough identities.

► **Definition 15.** *A function $t : A^{p^2} \rightarrow B$ is doubly cyclic if it satisfies every identity of the form $t(\mathbf{x}_1, \dots, \mathbf{x}_p) \approx t(\mathbf{y}_1, \dots, \mathbf{y}_p)$, where \mathbf{x}_i is a p -tuple of variables and \mathbf{y}_i is a cyclic shift of \mathbf{x}_i for every $i \in [p]$, and every identity of the form $t(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p) \approx t(\mathbf{x}_2, \dots, \mathbf{x}_p, \mathbf{x}_1)$, where each \mathbf{x}_i is a p -tuple of variables.*

Observe that t from Equation (1) is doubly cyclic – the first type of identities come from the cyclicity of the inner c while the second type from the outer c . It will be also useful for us to observe in Lemma 22 that, after rearranging the arguments (we read them row-wise), t is a cyclic function of arity p^2 . From the finiteness of the domain C we get one more property of function t . In the next definition, by an x/y -tuple we mean a tuple containing only variables x and y .

► **Definition 16.** *A doubly cyclic function $t : A^{p^2} \rightarrow B$ is b -bounded if there exists an equivalence relation \sim on the set of all p -ary x/y -tuples with at most b equivalence classes such that t satisfies every identity of the form $t(\mathbf{u}_1, \dots, \mathbf{u}_p) \approx t(\mathbf{v}_1, \dots, \mathbf{v}_p)$ where \mathbf{u}_i and \mathbf{v}_i are x/y -tuples such that $\mathbf{u}_i \sim \mathbf{v}_i$ for every $i \in [p]$.*

► **Lemma 17.** *Let $c : C^p \rightarrow C$ be a cyclic function. Then the function t defined by Equation (1) is a b -bounded doubly cyclic function for $b = |C|^{|C|^2}$.*

Proof. We define \sim by declaring two p -ary x/y -tuples \mathbf{u} and \mathbf{v} \sim -equivalent if $c(\mathbf{u}) \approx c(\mathbf{v})$. As there are $b = |C|^{|C|^2}$ binary functions $C^2 \rightarrow C$, this equivalence has at most b equivalence classes. By definitions, t is then b -bounded and doubly cyclic. ◀

The promised necessary condition for finite tractability is now a simple consequence:

► **Theorem 18.** *Let (\mathbb{A}, \mathbb{B}) be a finite PCSP template that is finitely tractable. Then there exists b such that (\mathbb{A}, \mathbb{B}) has a p^2 -ary b -bounded doubly cyclic polymorphism for every sufficiently large prime p .*

Proof. If (\mathbb{A}, \mathbb{B}) is finitely tractable, then, by Lemma 11, there exists a minion homomorphism $\xi : \text{Pol}(\mathbb{C}) \rightarrow \text{Pol}(\mathbb{A}, \mathbb{B})$, where \mathbb{C} is finite and $\text{CSP}(\mathbb{C})$ is tractable. By Theorem 14, \mathbb{C} has a p -ary cyclic polymorphism for every sufficiently large prime. Then, by Lemma 17, the polymorphism t of \mathbb{C} defined by Equation (1) is a b -bounded and doubly cyclic (with the appropriate b). As ξ preserves h1 identities, $\xi(t)$ is a b -bounded doubly cyclic polymorphism of (\mathbb{A}, \mathbb{B}) . ◀

3.3 Proof of Theorem 3

Finally, we are ready to start proving Theorem 3. Without loss of generality, we consider only templates on the first lines of Cases (1)–(3) of Theorem 3 (in particular, $r \leq s/2$) and assume that $r \leq s/2$ in Case (4) (the remaining templates can be obtained by swapping zero and one in the domains). We fix such a template (\mathbb{A}, \mathbb{B}) .

Striving for a contradiction, suppose that $\text{PCSP}(\mathbb{A}, \mathbb{B})$ is finitely tractable. By Theorem 18 there exists b such that (\mathbb{A}, \mathbb{B}) has a p^2 -ary b -bounded doubly cyclic polymorphism t for every sufficiently large arity p^2 . We fix such a b and t , where p is fixed to a sufficiently large prime p congruent to 1 modulo s (which is possible by the Dirichlet prime number theorem). How large must p be will be seen in due course. We denote $n = p^2$ and observe that $n \equiv 1 \pmod{s}$ as well.

Using the cyclicity in Section 4 and double cyclicity in Section 5 we will show that certain evaluations $t(\mathbf{z})$ of t are tame in that $t(\mathbf{z}) = t\langle 0 \rangle$ (recall here the notation in Subsection 2.3) if the area of \mathbf{z} is below a threshold θ . The *threshold* is defined as $\theta = 1/2$ for all the templates but the $(r$ -in- s , not-all-equal- s) template in Case (4), where we set $\theta = r/s$ (observe that $\theta = r/s$ also in Case (2) and (3)). We restate the definition of tameness for convenience.

► **Definition 19.** *A tuple $\mathbf{z} \in \{0, 1\}^n$ is tame if*

$$t(\mathbf{z}) = \begin{cases} t\langle 0 \rangle_n & \text{if } \lambda(\mathbf{z}) < \theta \\ 1 - t\langle 0 \rangle_n & \text{if } \lambda(\mathbf{z}) > \theta \end{cases}$$

(Note here that $\lambda(\mathbf{z})$ is never equal to θ since n is odd and $n \equiv 1 \pmod{s}$.)

The evaluations that we use are called near-threshold almost rectangles defined as follows.

► **Definition 20.** *A tuple $\mathbf{z} \in \{0, 1\}^n$ is an almost rectangle if it is a p -ary cyclic shift of a tuple of the form $\langle z^1, \dots, z^1, z^2, \dots, z^2 \rangle_p$, where $0 \leq z^1, z^2 \leq p$, the number of z^1 's is arbitrary, and $|z^1 - z^2| < 5b$. The quantity $\Delta z = |z^1 - z^2|$ is referred to as the step size. We say that \mathbf{z} is near-threshold if $|\lambda(\mathbf{z}) - \theta| < 1/s^{\Delta z+3}$.*

The proof can now be finished by using the tameness of near-threshold almost rectangles (that will be established in Lemma 25) together with the b -boundedness of t as follows.

Let $m = (p-1)/2$ and choose positive integers $z^{2,1}$ and $z^{2,2}$ so that $\theta p - 2b < z^{2,1} < z^{2,2} < \theta p$ and the x/y -tuples $(z^{2,1} \times x, (p - z^{2,1}) \times y)$ and $(z^{2,2} \times x, (p - z^{2,2}) \times y)$ are \sim -equivalent (see Definition 16 of boundedness). This is possible by the pigeonhole principle since there are more than b integers in the interval and \sim has at most b classes.

By the choice of $z^{2,1}$ and $z^{2,2}$, for any meaningful choice of z^1 , we have $t(\mathbf{z}_1) = t(\mathbf{z}_2)$ where $\mathbf{z}_i = \langle m \times z^1, (p - m) \times z^{2,i} \rangle_p$, $i = 1, 2$. We choose z^1 as the maximum number such that $\lambda(\mathbf{z}_1) < \theta$. (Note here that for $z^1 = p$ the area of \mathbf{z}_1 can be made arbitrarily close to

$(1 + \theta)/2 > \theta$ by choosing a sufficiently large p , so we may assume $z^1 < p$.) From $m < p/2$ it follows that increasing $z^{2,1}$ by one makes the area of \mathbf{z}_1 greater than increasing z^1 by one, therefore $\lambda(\mathbf{z}_2) > \theta$.

Note that $z^1 > p\theta$ since otherwise the area of \mathbf{z}_2 is less than θ . On the other hand, $z^1 < p\theta + 3b$, otherwise the area of \mathbf{z}_1 is greater (assuming $p > 5$):

$$\lambda(\mathbf{z}_1) = \frac{mz^1 + (p - m)z^{2,1}}{p^2} \geq \frac{\frac{p-1}{2}(p\theta + 3b) + \frac{p+1}{2}(p\theta - 2b)}{p^2} = \frac{p^2\theta + \frac{b(p-5)}{2}}{p^2} > \theta.$$

It follows that the step size of both \mathbf{z}_1 and \mathbf{z}_2 is less than $5b$, so both \mathbf{z}_i are almost rectangles. By choosing a sufficiently large p , the difference $\lambda(\mathbf{z}_2) - \lambda(\mathbf{z}_1)$ can be made arbitrarily small, and since $\lambda(\mathbf{z}_1) < \theta < \lambda(\mathbf{z}_2)$ both \mathbf{z}_i are then near-threshold.

Now the tameness of near-threshold almost rectangles (Lemma 25) gives us $t(\mathbf{z}_1) = t\langle 0 \rangle_n \neq 1 - t\langle 0 \rangle_n = t(\mathbf{z}_2)$. On the other hand, we also have $t(\mathbf{z}_1) = t(\mathbf{z}_2)$, a contradiction.

4 Step size at most one

In this section we prove the following lemma.

► **Lemma 21.** *Every near-threshold almost rectangle of step size at most one is tame.*

We will use the cyclicity of an operation obtained from t by an appropriate rearrangement of its arguments, stated in the following lemma. Its proof is in Appendix A.

► **Lemma 22.** *Let $t : A^{p^2} \rightarrow B$ be a doubly cyclic function. Then the function t^σ defined by*

$$t^\sigma \left(\begin{array}{cccc} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pp} \end{array} \right) = t \left(\begin{array}{cccc} x_{11} & x_{21} & \cdots & x_{p1} \\ x_{12} & x_{22} & \cdots & x_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{pp} \end{array} \right)$$

is a cyclic function.

Observe that an almost rectangle $\mathbf{z} = \langle z^2 + 1, \dots, z^2 + 1, z^2, \dots, z^2 \rangle_p$ regarded as a $p \times p$ matrix is, when read row-wise, equal to a sequence of consecutive ones, followed by zeros. In other words, using the notation t^σ from Lemma 22, we have $t(\mathbf{z}) = t^\sigma \langle k \rangle_n$ for some k . Also note that every almost rectangle of step size at most one has a p -ary cyclic shift of this form. Finally, notice that if \mathbf{z} is near-threshold, then $k \leq 2\lfloor \theta n \rfloor$. In order to prove Lemma 21, it is therefore enough to verify the following lemma.

► **Lemma 23.** *Denote $a = \lfloor \theta n \rfloor$. For every $0 \leq k \leq 2a$, we have*

$$t^\sigma \langle k \rangle_n = \begin{cases} t^\sigma \langle 0 \rangle_n & \text{if } 0 \leq k \leq a \\ 1 - t^\sigma \langle 0 \rangle_n & \text{if } 1 + a \leq k \leq 2a \end{cases}$$

The rest of this section is devoted to proving this lemma. We require an additional definition. We say that an s -tuple of evaluations $\langle k_1 \rangle_n, \dots, \langle k_s \rangle_n$, where $0 \leq k_i \leq n$, is *plausible* if $\sum_{i=1}^s k_i = rn$ in Cases (1), (3), (4) and $\sum_{i=1}^s k_i \leq rn$ in Case (2). The following lemma is a consequence of the fact that t^σ is a polymorphism (as t is) which is, additionally, cyclic by Lemma 22. No other properties of t are needed in this section.

► **Lemma 24.** *If an s -tuple $\langle k_1 \rangle_n, \dots, \langle k_s \rangle_n$ is plausible, then $(t^\sigma \langle k_1 \rangle_n, \dots, t^\sigma \langle k_s \rangle_n) \in Q$ (recall here that (P, Q) is introduced in the statement of Theorem 3).*

Moreover, in Cases (1), (2), and (3), we have $t^\sigma \langle n - k \rangle_n = 1 - t^\sigma \langle k \rangle_n$ for every $0 \leq k \leq n$.

11:10 Finitely Tractable Promise Constraint Satisfaction Problems

Proof. For the first part, let $\langle k_1 \rangle, \dots, \langle k_s \rangle$ be plausible. Form an $s \times rn$ matrix M whose first row is $\langle k_1 \rangle_{rn}$ and the j -th row is the $(\sum_{l=1}^{j-1} k_l)$ -th cyclic shift of $\langle k_j \rangle_{rn}$ for $j \in \{2, \dots, s\}$. Note that each of the first $\sum k_i$ columns of M contains exactly 1 one and the remaining columns are all zero (the latter only applies in Case (2)). Split this matrix into r -many $s \times n$ blocks M^1, M^2, \dots, M^r . Their sum $X = \sum_{j=1}^r M^j$ is an $s \times n$ zero-one matrix whose each column contains exactly r ones in Cases (1), (3), and (4), and at most r ones in Case (2). Moreover, for all $j \in [s]$, the j -th row of X is a cyclic shift of $\langle k_j \rangle$, therefore its t^σ -image is $t^\sigma \langle k_j \rangle$ by cyclicity of t^σ . Each column belongs to the relation P , therefore, as t^σ is a polymorphism, we get that t^σ applied to the rows gives a tuple in Q . This implies the first claim.

For the second part, we take $\langle k \rangle$ together with the k -th cyclic shift of $\langle n - k \rangle$ and use the fact that t^σ preserves the disequality relation pair. ◀

We now consider Cases (1)–(4) separately. Case (2) is the simplest. If $0 \leq k \leq a$ then $\langle k \rangle, \langle k \rangle, \dots, \langle k \rangle$ is a plausible tuple. By Lemma 24, the tuple $(t^\sigma \langle k \rangle, t^\sigma \langle k \rangle, \dots, t^\sigma \langle k \rangle)$ is in Q ; therefore $t^\sigma \langle k \rangle = 0$. For the remaining values $2a \geq k \geq a + 1$ we apply the second part of this lemma and get $t^\sigma \langle k \rangle = 1$.

For Case (1) we prove $t^\sigma \langle k \rangle = 0$ and $t^\sigma \langle n - k \rangle = 1$ for any $0 \leq k \leq a$ by induction on $i = a - k$, $i = 0, 1, \dots, a$. For the first step, $k = (n - 1)/2$, we apply Lemma 24 to the plausible s -tuple $2r \times \langle k \rangle, \langle r \rangle, (s - 2r - 1) \times \langle 0 \rangle$. Since Q contains no p -tuple with more than $(2r - 1)$ ones, we get $t^\sigma \langle k \rangle = 0$. Then also $t^\sigma \langle n - k \rangle = 1$ by the second part of the lemma. For the induction step, we use the tuple

$$r \times \langle k \rangle, r \times \langle n - k - 1 \rangle, \langle r \rangle, (s - 2r - 1) \times \langle 0 \rangle$$

in a similar way, additionally using that $t^\sigma \langle n - k - 1 \rangle = 1$ by the induction hypothesis.

We proceed to Case (4). We will prove, starting from the left, the following chain of disequalities.

$$t^\sigma \langle a \rangle \neq t^\sigma \langle a + 1 \rangle \neq t^\sigma \langle a - 1 \rangle \neq t^\sigma \langle a + 2 \rangle \neq t^\sigma \langle a - 2 \rangle \neq \dots \neq t^\sigma \langle 2a \rangle \neq t^\sigma \langle 0 \rangle$$

This will imply $t^\sigma \langle a \rangle = t^\sigma \langle a - 1 \rangle = \dots = t^\sigma \langle 0 \rangle \neq t^\sigma \langle a + 1 \rangle = t^\sigma \langle a + 2 \rangle = \dots = t^\sigma \langle 2a \rangle$. We start with the first disequality $t^\sigma \langle a \rangle \neq t^\sigma \langle a + 1 \rangle$. The sequence of arguments

$$(s - r) \times \langle a \rangle, r \times \langle a + 1 \rangle$$

has length s and is plausible as $(s - r)a + r(a + 1) = sa + r$ and $sa + r$ is equal to rn . (Indeed, $n \equiv 1 \pmod{s}$, so $n = ms + 1$ for some integer m ; then $a = mr$ and $sa + r = smr + r = (n - 1)r + r = rn$.) By Lemma 24, $t^\sigma \langle a \rangle \neq t^\sigma \langle a + 1 \rangle$ since Q does not contain all-equal tuples in Case (4). The remaining disequalities are proved in Appendix B.

Case (3) can be done similarly as Case (4) with an additional reasoning that we now explain. Consider, e.g., the proof that $t^\sigma \langle a \rangle \neq t^\sigma \langle a + 1 \rangle$ using the sequence $(s - r) \times \langle a \rangle, r \times \langle a + 1 \rangle$. We cannot directly conclude that $t^\sigma \langle a \rangle \neq t^\sigma \langle a + 1 \rangle$ since relation Q contains the all-zero tuple – we can only conclude that $t^\sigma \langle a \rangle$ and $t^\sigma \langle a + 1 \rangle$ are not both ones. However, we can also prove in the same way that $t^\sigma \langle n - a \rangle$ and $t^\sigma \langle n - (a + 1) \rangle$ are not both ones by using the “complementary” tuple $(s - r) \times \langle n - a \rangle, r \times \langle n - (a + 1) \rangle$. The claim $t^\sigma \langle a \rangle \neq t^\sigma \langle a + 1 \rangle$ then follows from the second part of Lemma 24.

The proof of Lemma 23 is concluded.

5 Arbitrary step size

The entire section is devoted to the proof of the following lemma.

► **Lemma 25.** *Every near-threshold almost rectangle is tame.*

We start by redefining plausibility.

We say that an m -tuple of evaluations $\mathbf{k}_1 = \langle k_1^1, \dots, k_1^p \rangle, \dots, \mathbf{k}_m = \langle k_m^1, \dots, k_m^p \rangle$, where $m \in [s]$, is *plausible* if $\sum_{j=1}^m k_j^i = rp$ for all $i \in [p]$ (note that we do not make exception for Case (2) here). In other words, by arranging the integers defining $\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_m$ as rows of an $m \times p$ matrix, we get a matrix whose every column sums up to rp . Note that the sum of the areas of the evaluations is then equal to r .

The following lemma is a “2-dimensional analogue” of Lemma 24. The proof applies the first type of doubly cyclic identities from Definition 15, it is given in Appendix C.

► **Lemma 26.** *If a tuple $\mathbf{k}_1, \dots, \mathbf{k}_s$ is plausible, then $(t(\mathbf{k}_1), \dots, t(\mathbf{k}_s)) \in Q$.*

Moreover, in Cases (1), (2), and (3), we have $t(p - k^1, \dots, p - k^p) = 1 - t(k^1, \dots, k^p)$ for any evaluation $\langle k^1, \dots, k^p \rangle$.

The next lemma will be applied to produce plausible sequence of evaluations. The proof uses the other type of doubly cyclic identities. It is given in Appendix D, here we provide a brief sketch. (In the statement, note that $r/\theta = s$ except for Case (1) where $r/\theta = 2r$.)

► **Lemma 27.** *Let \mathbf{z} be an almost rectangle of step size $\Delta z \geq 2$ with $|\lambda(\mathbf{z}) - \theta| \leq 1/s^3$ and let p be sufficiently large. Then*

- *there exists a plausible r/θ -tuple $\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_{r/\theta-1}, \mathbf{l}$ of almost rectangles such that $t(\mathbf{z}) = t(\mathbf{k}_1) = t(\mathbf{k}_2) = \dots = t(\mathbf{k}_{r/\theta-1}), \lambda(\mathbf{z}) = \lambda(\mathbf{k}_1) = \dots = \lambda(\mathbf{k}_{r/\theta-1})$, and \mathbf{l} has the same step size Δz as \mathbf{z} ;*
- *there exists a plausible r/θ -tuple $\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_{r/\theta-2}, \mathbf{l}_1, \mathbf{l}_2$ of almost rectangles such that $t(\mathbf{z}) = t(\mathbf{k}_1) = t(\mathbf{k}_2) = \dots = t(\mathbf{k}_{r/\theta-2}), \lambda(\mathbf{z}) = \lambda(\mathbf{k}_1) = \dots = \lambda(\mathbf{k}_{r/\theta-2})$, both \mathbf{l}_1 and \mathbf{l}_2 have step size strictly smaller than Δz , and $|\lambda(\mathbf{l}_1) - \lambda(\mathbf{l}_2)| \leq 1/p$.*

Proof sketch. We can assume that $\mathbf{z} = \langle c \times z^1, d \times z^2 \rangle$ for some c, d, z^1, z^2 . For the first item, we consider the $(r/\theta - 1) \times p$ matrix X whose first row is \mathbf{z} and the i -th row is the c -th cyclic shift of the $(i - 1)$ -st row for each $i \in \{2, \dots, r/\theta - 1\}$. Let Y be the $r/\theta \times p$ matrix obtained from X by adding a row (l^1, \dots, l^p) so that each column sums up to rp and we define $\mathbf{k}_1, \dots, \mathbf{k}_m, \mathbf{l}$ as the n -tuples determined by the rows of Y via $\langle \rangle$, e.g., $\mathbf{l} = \langle l^1, \dots, l^p \rangle$. The inequality $|\lambda(\mathbf{z}) - \theta| \leq 1/s^3$ (and p being sufficiently large) ensures that \mathbf{l} is correctly defined (i.e., all the l^i are between 0 and p), the construction gives that \mathbf{l} is an almost rectangle with step size Δz and that \mathbf{z} and \mathbf{k}_i have equal areas, and the double cyclicity of t implies $t(\mathbf{z}) = t(\mathbf{k}_i)$. For the second item we additionally split the l row in two roughly equal rows. This will guarantee the two properties of \mathbf{l}_1 and \mathbf{l}_2 . ◀

Equipped with these lemmata we are ready to prove Lemma 25. The proof is by induction on the step size. Step sizes zero and one are dealt with in Lemma 21, so we assume that \mathbf{z} is a near-threshold almost rectangle of step size $2 \leq \Delta z < 5b$.

We will consider Case (4) in detail and discuss the adjustments for the other cases afterwards. Assume first that $\lambda(\mathbf{z})$ is not *too close to* θ , say, $|\lambda(\mathbf{z}) - \theta| \geq 1/s^{5b+4}$. We apply the second item in Lemma 27 and get a plausible s -tuple $\mathbf{k}_1, \dots, \mathbf{k}_{s-2}, \mathbf{l}_1, \mathbf{l}_2$ such that $\mathbf{z}, \mathbf{k}_1, \dots, \mathbf{k}_{s-2}$ all have the same t -images and areas, and \mathbf{l}_1 and \mathbf{l}_2 are almost rectangles with step sizes strictly smaller than Δz , whose areas differ by at most $1/p$.

11:12 Finitely Tractable Promise Constraint Satisfaction Problems

The average area of almost rectangles $\mathbf{k}_1, \dots, \mathbf{k}_{s-2}, \mathbf{l}_1, \mathbf{l}_2$ is $r/s = \theta$, the first $s - 2$ of them have the same area as \mathbf{z} , bounded away from θ by a constant (namely $1/s^{5b+4}$), and the last two have almost the same area (the difference is at most $1/p$). By choosing a large enough p we get $\text{sgn}(\lambda(\mathbf{l}_1) - \theta) = \text{sgn}(\lambda(\mathbf{l}_2) - \theta) \neq \text{sgn}(\lambda(\mathbf{z}) - \theta)$ and $|\lambda(\mathbf{l}_i) - \theta| \leq s \cdot |\lambda(\mathbf{z}) - \theta|$; in particular, both \mathbf{l}_i are near-threshold since $s \cdot |\lambda(\mathbf{z}) - \theta| \leq 1/s^{\Delta z+3-1} \leq 1/s^{\Delta l_i+3}$. By the induction hypothesis, both \mathbf{l}_i are tame. By Lemma 26, the values $t(\mathbf{k}_1), \dots, t(\mathbf{k}_{s-2}), t(\mathbf{l}_1)$, and $t(\mathbf{l}_2)$ are not all equal. But $t(\mathbf{z}) = t(\mathbf{k}_1) = \dots = t(\mathbf{k}_{s-2}), t(\mathbf{l}_1) = t(\mathbf{l}_2)$, and $\text{sgn}(\lambda(\mathbf{z}) - \theta) \neq \text{sgn}(\lambda(\mathbf{l}_1) - \theta)$ so it follows that \mathbf{z} is tame, as required.

It remains to deal with the case that $\lambda(\mathbf{z})$ is too close to θ . In this case we will find an almost rectangle \mathbf{l} with the same step size as \mathbf{z} such that $t(\mathbf{l}) = 1 - t(\mathbf{z})$ and $\lambda(\mathbf{l}) - \theta = -s'(\lambda(\mathbf{z}) - \theta)$, where s' is such that $2 \leq s' \leq s$. If $\lambda(\mathbf{l})$ is already not too close to the threshold θ , then we observe that \mathbf{l} is near-threshold (indeed, $|\lambda(\mathbf{l}) - \theta| \leq s|\lambda(\mathbf{z}) - \theta| \leq s/s^{5b+4} \leq 1/s^{\Delta z+3}$) and apply to \mathbf{l} the first part of the proof, thus obtaining that \mathbf{l} is tame and, consequently, \mathbf{z} is tame as well. If $\lambda(\mathbf{l})$ is still too close to θ , then we simply repeat the process until we get a rectangle that is not too close.

To find such an almost rectangle \mathbf{l} we apply the first item of Lemma 27 and get a plausible s -tuple $\mathbf{k}_1, \dots, \mathbf{k}_{s-1}, \mathbf{l}$ such that $t(\mathbf{z}) = t(\mathbf{k}_1) = \dots = t(\mathbf{k}_{s-1})$ and \mathbf{l} is an almost rectangle of the same step size as \mathbf{z} . Since the area of each \mathbf{k}_i is equal to $\lambda(\mathbf{z})$ and the average area in the plausible s -tuple is θ , we get that $\lambda(\mathbf{l}) - \theta = -(s - 1)(\lambda(\mathbf{z}) - \theta)$. By Lemma 26, $t(\mathbf{l})$ and $t(\mathbf{z})$ are not equal. This concludes the construction of \mathbf{l} and the proof of Lemma 25 for Case (4).

The remaining cases (1), (2), and (3) require a modification that is similar to the modification for Case (3) in the proof of Lemma 23. Consider the situation that $\lambda(\mathbf{z})$ is not too close to θ . In Cases (2) and (3) Lemma 27 is applied not only to $\mathbf{k}_1, \dots, \mathbf{k}_{2r-2}, \mathbf{l}_1, \mathbf{l}_2$ but also to the tuple formed by “complementary” almost rectangles, which have different t -images by the second part of Lemma 26. In Case (1) we additionally complete the two $2r$ tuples to s -tuples by adding $s - 2r$ zeros. The other situation, that $\lambda(\mathbf{z})$ is too close, is adjusted in an analogous fashion.

6 Conclusion

We have characterized finite tractability among the basic tractable cases in the Brakensiek–Guruswami classification [8] of symmetric Boolean PCSPs allowing negations. A natural direction for future research is an extension to all the tractable cases (not just the basic ones), or even to all symmetric Boolean PCSPs [14], not only those allowing negations. An obstacle, where our efforts have failed so far, is already in relaxations of the basic templates (P, Q) with disequalities. For example, which (P, Q) , (\neq, \neq) , with P a subset of $\leq r$ -in- s and Q a superset of $\leq (2r - 1)$ -in- s , give rise to finitely tractable PCSPs?

Another natural direction is to better understand the “level of tractability.” For the finitely tractable templates (\mathbb{A}, \mathbb{B}) considered in this paper, it is always possible to find a tractable CSP (\mathbb{C}) with $\mathbb{A} \rightarrow \mathbb{C} \rightarrow \mathbb{B}$ and such that \mathbb{C} is two-element. Is it so for all symmetric Boolean templates? For general Boolean templates, the answer is “No”: [12] presents an example that requires a three-element \mathbb{C} . However, it is unclear whether there is an upper bound on the size of \mathbb{C} for finitely tractable (Boolean) PCSPs, and if there is, how it could be computed. There are also natural concepts beyond finite tractability, still stronger than standard tractability. We refer to [2] for some questions in this direction.

References

- 1 Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2 + \epsilon)$ -Sat is NP-hard. *SIAM J. Comput.*, 46(5):1554–1573, 2017. doi:10.1137/15M1006507.
- 2 L. Barto. Promises make finite (constraint satisfaction) problems infinitary. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–8, 2019.
- 3 Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Oprsal. Algebraic approach to promise constraint satisfaction. *CoRR*, abs/1811.00970, 2018. arXiv:1811.00970.
- 4 Libor Barto and Marcin Kozik. Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem. *Log. Methods Comput. Sci.*, 8(1:07):1–26, 2012. Special issue: Selected papers of the Conference “Logic in Computer Science (LICS) 2010”. doi:10.2168/LMCS-8(1:07)2012.
- 5 Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and how to use them. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol17.15301.1.
- 6 Libor Barto, Jakub Opršal, and Michael Pinsker. The wonderland of reflections. *Israel Journal of Mathematics*, 223(1):363–398, February 2018. doi:10.1007/s11856-017-1621-9.
- 7 Manuel Bodirsky. Constraint satisfaction problems with infinite templates. In Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints*, volume 5250 of *Lecture Notes in Computer Science*, pages 196–228. Springer, 2008. doi:10.1007/978-3-540-92800-3_8.
- 8 Joshua Brakensiek and Venkatesan Guruswami. Promise constraint satisfaction: Structure theory and a symmetric boolean dichotomy. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA’18, pages 1782–1801, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=3174304.3175422>.
- 9 Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.
- 10 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, October 2017. doi:10.1109/FOCS.2017.37.
- 11 Jakub Bulín, Andrei Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC ’19)*, New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316300.
- 12 Guofeng Deng, Ezzeddine El Sai, Trevor Manders, Peter Mayr, Poramate Nakkirt, and Athena Sparks. Sandwiches for promise constraint satisfaction, 2020. arXiv:2003.07487.
- 13 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, February 1998. doi:10.1137/S0097539794266766.
- 14 Miron Fícak, Marcin Kozik, Miroslav Olsák, and Szymon Stankiewicz. Dichotomy for Symmetric Boolean PCSPs. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:12, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.57.
- 15 Pavol Hell and Jaroslav Nešetřil. On the complexity of H -coloring. *J. Combin. Theory Ser. B*, 48(1):92–110, 1990.
- 16 Peter Jeavons. On the algebraic structure of combinatorial problems. *Theor. Comput. Sci.*, 200(1-2):185–204, 1998.
- 17 Vladimir Kolmogorov, Andrei Krokhin, and Michal Rolínek. The complexity of general-valued CSPs. *SIAM Journal on Computing*, 46(3):1087–1110, 2017.

- 18 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, STOC '78*, pages 216–226, New York, NY, USA, 1978. ACM. doi:10.1145/800133.804350.
- 19 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 331–342, October 2017. doi:10.1109/FOCS.2017.38.
- 20 Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5), 2020. doi:10.1145/3402029.

A

 Doubly cyclic functions are cyclic

In this appendix we prove Lemma 22, which we restate here for convenience.

► **Lemma 22.** *Let $t : A^{p^2} \rightarrow B$ be a doubly cyclic function. Then the function t^σ defined by*

$$t^\sigma \left(\begin{array}{cccc} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pp} \end{array} \right) = t \left(\begin{array}{cccc} x_{11} & x_{21} & \cdots & x_{p1} \\ x_{12} & x_{22} & \cdots & x_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{pp} \end{array} \right)$$

is a cyclic function.

Proof. By cyclically shifting the arguments we get the same result:

$$\begin{aligned} & t^\sigma(x_{21}, x_{31}, \dots, x_{p1}, x_{12}, x_{22}, x_{32}, \dots, x_{p2}, x_{13}, \dots, x_{2p}, x_{3p}, \dots, x_{pp}, x_{11}) \\ &= t^\sigma \left(\begin{array}{cccc} x_{21} & \cdots & x_{2,p-1} & x_{2p} \\ \vdots & \ddots & \vdots & \vdots \\ x_{p1} & \cdots & x_{p,p-1} & x_{pp} \\ x_{12} & \cdots & x_{1p} & x_{11} \end{array} \right) = t \left(\begin{array}{cccc} x_{21} & \cdots & x_{p1} & x_{12} \\ \vdots & \ddots & \vdots & \vdots \\ x_{2,p-1} & \cdots & x_{p,p-1} & x_{1p} \\ x_{2p} & \cdots & x_{pp} & x_{11} \end{array} \right) \\ &= t \left(\begin{array}{cccc} x_{21} & \cdots & x_{p1} & x_{11} \\ \vdots & \ddots & \vdots & \vdots \\ x_{2,p-1} & \cdots & x_{p,p-1} & x_{1,p-1} \\ x_{2p} & \cdots & x_{pp} & x_{1p} \end{array} \right) = t \left(\begin{array}{cccc} x_{11} & x_{21} & \cdots & x_{p1} \\ x_{12} & x_{22} & \cdots & x_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{pp} \end{array} \right) \\ &= t^\sigma \left(\begin{array}{cccc} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pp} \end{array} \right) \\ &= t^\sigma(x_{11}, x_{21}, \dots, x_{p1}, x_{12}, x_{22}, \dots, x_{p2}, \dots, x_{1p}, x_{2p}, \dots, x_{pp}). \quad \blacktriangleleft \end{aligned}$$

B

 Step size one, Case (4)

In this section we finish the proof of Lemma 23 for Case (4). We state the lemma for convenience.

► **Lemma 23.** *Denote $a = \lfloor \theta n \rfloor$. For every $0 \leq k \leq 2a$, we have*

$$t^\sigma \langle k \rangle_n = \begin{cases} t^\sigma \langle 0 \rangle_n & \text{if } 0 \leq k \leq a \\ 1 - t^\sigma \langle 0 \rangle_n & \text{if } 1 + a \leq k \leq 2a \end{cases}$$

Recall that we want to prove, starting from the left, the chain of disequalities

$$t^\sigma \langle a \rangle \neq t^\sigma \langle a+1 \rangle \neq t^\sigma \langle a-1 \rangle \neq t^\sigma \langle a+2 \rangle \neq t^\sigma \langle a-2 \rangle \neq \dots \neq t^\sigma \langle 2a \rangle \neq t^\sigma \langle 0 \rangle$$

and that we have already verified the first one.

For the second disequality $t^\sigma \langle a+1 \rangle \neq t^\sigma \langle a-1 \rangle$, as well as for the further disequalities we need to distinguish two cases: Case (4a) r and s have the same parity and Case (4b) r is even and s is odd. In Case (4a) we directly use the sequence

$$(s-r)/2 \times \langle a-1 \rangle, (s+r)/2 \times \langle a+1 \rangle$$

and derive $t^\sigma \langle a+1 \rangle \neq t^\sigma \langle a-1 \rangle$ using Lemma 24 as before. In Case (4b) we first use

$$(s-1) \times \langle a \rangle, \langle a+r \rangle$$

to deduce $t^\sigma \langle a+r \rangle \neq t^\sigma \langle a \rangle$ (so $t^\sigma \langle a+1 \rangle = t^\sigma \langle a+r \rangle$) and then

$$(s-1)/2 \times \langle a-1 \rangle, (s-1)/2 \times \langle a+1 \rangle, \langle a+r \rangle$$

to deduce $t^\sigma \langle a-1 \rangle \neq t^\sigma \langle a+1 \rangle$.

To prove $t^\sigma \langle a-i+1 \rangle \neq t^\sigma \langle a+i \rangle$ for $i \in \{2, 3, \dots, a\}$, we observe that, by the already established disequalities, we have $t^\sigma \langle a-i+1 \rangle = \dots = t^\sigma \langle a \rangle$, and then use

- $(s+r)/4 \times \langle a+i \rangle, (s-r)/2 \times \langle a-1 \rangle, (s+r)/4 \times \langle a-i+2 \rangle$ in Case (4a) and $(s+r)/2$ is even;
- $(s+r+2)/4 \times \langle a+i \rangle, (s-r-2)/2 \times \langle a-1 \rangle, 2 \times \langle a-i+1 \rangle, (s+r-6)/4 \times \langle a-i+2 \rangle$ in Case (4a) and $(s+r)/2$ is odd;
- $r/2 \times \langle a+i \rangle, (s-r) \times \langle a \rangle, r/2 \times \langle a-i+2 \rangle$ in Case (4b).

Finally, for proving $t^\sigma \langle a+i \rangle \neq t^\sigma \langle a-i \rangle$ we use

- $(s-r)/2 \times \langle a-i \rangle, (s-r)/2 \times \langle a+i \rangle, r \times \langle a+1 \rangle$ in Case (4a) and
- $(s-1)/2 \times \langle a-i \rangle, (s-1)/2 \times \langle a+i \rangle, 1 \times \langle a+r \rangle$ in Case (4b).

This completes the proof for Case (4).

C Proof of Lemma 26

► **Lemma 26.** *If a tuple $\mathbf{k}_1, \dots, \mathbf{k}_s$ is plausible, then $(t(\mathbf{k}_1), \dots, t(\mathbf{k}_s)) \in Q$.*

Moreover, in Cases (1), (2), and (3), we have $t(p-k^1, \dots, p-k^p) = 1 - t(k^1, \dots, k^p)$ for any evaluation $\langle k^1, \dots, k^p \rangle$.

Proof. Let $\mathbf{k}_1, \dots, \mathbf{k}_s$ be a plausible tuple. Fix, for a while, an arbitrary $i \in [p]$. Form a $s \times rp$ matrix M_i whose first row is $\langle k_1^i \rangle_{rp}$ and j -th row is the $(\sum_{l=1}^{j-1} k_l^i)$ -th cyclic shift of $\langle k_j^i \rangle_{rp}$ for $j \in \{2, \dots, s\}$. Split this matrix into r -many $s \times p$ blocks $M_i^1, M_i^2, \dots, M_i^r$. Their sum $X_i = \sum_{j=1}^r M_i^j$ is an $s \times p$ matrix whose each column contains exactly r ones. Moreover, for all $j \in [s]$, the j -th row of the matrix X_i is a cyclic shift of $\langle k_j^i \rangle_p$. Put the matrices X_1, \dots, X_p aside to form an $s \times n$ matrix Y . Its rows have the same t -images as $\mathbf{k}_1, \dots, \mathbf{k}_s$, respectively, because t is doubly cyclic. Each column belongs to the relation P , therefore, as t is a polymorphism, we get that t applied to the rows gives a tuple in Q . This tuple is equal to $(t(\mathbf{k}_1), \dots, t(\mathbf{k}_s))$.

The second part can be proved in a similar way as the second part of Lemma 24 using the disequality relation pair. ◀

D Proof of Lemma 27

► **Lemma 27.** *Let \mathbf{z} be an almost rectangle of step size $\Delta z \geq 2$ with $|\lambda(\mathbf{z}) - \theta| \leq 1/s^3$ and let p be sufficiently large. Then*

- *there exists a plausible r/θ -tuple $\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_{r/\theta-1}, \mathbf{l}$ of almost rectangles such that $t(\mathbf{z}) = t(\mathbf{k}_1) = t(\mathbf{k}_2) = \dots = t(\mathbf{k}_{r/\theta-1}), \lambda(\mathbf{z}) = \lambda(\mathbf{k}_1) = \dots = \lambda(\mathbf{k}_{r/\theta-1}),$ and \mathbf{l} has the same step size Δz as \mathbf{z} ;*
- *there exists a plausible r/θ -tuple $\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_{r/\theta-2}, \mathbf{l}_1, \mathbf{l}_2$ of almost rectangles such that $t(\mathbf{z}) = t(\mathbf{k}_1) = t(\mathbf{k}_2) = \dots = t(\mathbf{k}_{r/\theta-2}), \lambda(\mathbf{z}) = \lambda(\mathbf{k}_1) = \dots = \lambda(\mathbf{k}_{r/\theta-2}),$ both \mathbf{l}_1 and \mathbf{l}_2 have step size strictly smaller than $\Delta z,$ and $|\lambda(\mathbf{l}_1) - \lambda(\mathbf{l}_2)| \leq 1/p.$*

Proof. Without loss of generality we can assume that $\mathbf{z} = \langle c \times z^1, d \times z^2 \rangle$ for some c, d and $z^1 > z^2$. Let $m = r/\theta - 1$ for the first item and $m = r/\theta - 2$ for the second one. We define an integer $m \times p$ matrix X so that the first row is $(c \times z^1, d \times z^2)$ and the i -th row is the c -th cyclic shift of the $(i - 1)$ -st row for each $i \in \{2, \dots, m\}$. Let Y be the $(m + 1) \times p$ matrix obtained from X by adding a row (l^1, \dots, l^p) so that each column sums up to rp . It is easily seen by induction on $i \leq m$ that the sum of the first i rows is a cyclic shift of a tuple of the form $(e, \dots, e, e', \dots, e')$, where $|e - e'| = \Delta z$ and the “step down” is at position $ci \pmod p$ (when columns are indexed from 0). It follows that (l^1, \dots, l^p) is also a cyclic shift of a tuple of the form $(e, \dots, e, e', \dots, e')$ where e and e' differ by Δz .

Next we observe that each $l^i > 0$ if p is sufficiently large. Indeed, note that since $|z^1 - z^2|/p$ can be made arbitrarily small (recall $|z^1 - z^2| < 5b$), we have $p(\lambda(\mathbf{z}) - \epsilon) < z^1, z^2 < p(\lambda(\mathbf{z}) + \epsilon)$, where $\epsilon > 0$ can be made arbitrarily small. We then have $l^i > rp - mp(\lambda(\mathbf{z}) + \epsilon) \geq rp - (r/\theta - 1)p(\theta + 1/s^3 + \epsilon) = p(\theta - (r/\theta - 1)(1/s^3 + \epsilon)) > p(\theta - r/\theta(1/s^3 + \epsilon))$, which is, for a sufficiently small ϵ , greater than 0 since $r/\theta s^3 \leq 1/s^2 < \theta$. Similarly, each $l^i < 2\theta \leq p$ if $m = r/\theta - 1$ and $l^i < 3\theta$ if $m = r/\theta - 2$.

Now we can finish the proof of the first item. We set $\mathbf{k}_1, \dots, \mathbf{k}_m, \mathbf{l}$ to be the n -tuples determined by the rows of Y via $\langle \rangle$, e.g., $\mathbf{l} = \langle l^1, \dots, l^p \rangle$. The inequalities $0 \leq l^i \leq p$ guarantee that \mathbf{l} is correctly defined and we see, using also the double cyclicity of t (for $t(\mathbf{z}) = t(\mathbf{k}_1) = \dots$), that these n -tuples have all the required properties.

To finish the proof of the second item, we define the \mathbf{k}_i as above and set $\mathbf{l}_1 = \langle \lfloor l^1/2 \rfloor, \dots, \lfloor l^p/2 \rfloor \rangle, \mathbf{l}_2 = \langle \lceil l^1/2 \rceil, \dots, \lceil l^p/2 \rceil \rangle$. Since $0 \leq l^i \leq 3\theta/2 < p$, these tuples are correctly defined almost rectangles. Their areas clearly differ by at most $1/p$. As $\Delta z \geq 2$, their step sizes are strictly smaller than Δz , and we are done in this case as well. ◀

A Generic Strategy Improvement Method for Simple Stochastic Games

David Auger ✉

Université Paris Saclay, UVSQ, DAVID, France

Xavier Badin de Montjoye ✉

Université Paris Saclay, UVSQ, DAVID, France

Yann Strozecki ✉

Université Paris Saclay, UVSQ, DAVID, France

Abstract

We present a generic strategy improvement algorithm (GSIA) to find an optimal strategy of simple stochastic games (SSG). We prove the correctness of GSIA, and derive a general complexity bound, which implies and improves on the results of several articles. First, we remove the assumption that the SSG is stopping, which is usually obtained by a polynomial blowup of the game. Second, we prove a tight bound on the denominator of the values associated to a strategy, and use it to prove that all strategy improvement algorithms are in fact fixed parameter tractable in the number r of random vertices. All known strategy improvement algorithms can be seen as instances of GSIA, which allows to analyze the complexity of converge from below by Condon [14] and to propose a class of algorithms generalising Gimbert and Horn’s algorithm [16, 17]. These algorithms terminate in at most $r!$ iterations, and for binary SSGs, they do less iterations than the current best deterministic algorithm given by Ibsen-Jensen and Miltersen [18].

2012 ACM Subject Classification Theory of computation → Algorithmic game theory

Keywords and phrases Simple Stochastic Games, Strategy Improvement, Parametrized Complexity, Stopping, Meta Algorithm, f-strategy

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.12

Related Version *Full Version*: <https://arxiv.org/abs/2102.04922>

Acknowledgements The authors want to thank Pierre Coucheney for many interesting discussions on SSGs.

1 Introduction

A *simple stochastic game*, or SSG, is a two-player turn-based zero-sum game with perfect information introduced by Condon [13]. It is a simpler version of *stochastic games*, previously defined by Shapley [23]. An SSG is played by two players MAX and MIN moving a pebble on a graph. Vertices are divided into MIN vertices, MAX vertices, random vertices and a target vertex for MAX. When the pebble reaches a MIN or MAX vertex, corresponding players move the pebble to a neighbouring vertex of their choice. If it reaches a random vertex, the next vertex is chosen at random following some probability law. Finally, when the pebble reaches the target vertex, MIN pays 1 to MAX. The goal of MIN is to minimise the probability to reach the target vertex while MAX must maximise this probability.

We study the algorithmic problem of *solving* an SSG, i.e. finding a pair of optimal strategies in an SSG, or equivalently the optimal value vector of the optimal probabilities for MAX to reach the sink from each vertex. There are always optimal strategies for both players that are positional [13], i.e. stationary and deterministic, but the number of positional strategies is exponential in the size of the game. Consequently, finding a pair of optimal strategies is a problem not known to be in FP, but it is in PPAD [20], a class included in FNP.



© David Auger, Xavier Badin de Montjoye, and Yann Strozecki;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 12; pp. 12:1–12:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Simple Stochastic Games can be used to simulate many classical games such as parity games, mean or discounted payoff games [2, 9]. Moreover, stochastic versions of these games are equivalent to SSGs [2], which underlines that SSGs are an important model to study. SSGs have applications in different domains such as model checking of modal μ -calculus [24], or modelling autonomous urban driving [11].

There are roughly three known methods to solve SSGs: strategy improvement, value iteration and quadratic programming. A strategy improvement algorithm (SIA) starts with a strategy for one player and improves it until it is optimal, whereas value iteration algorithms (VIA) update a value vector by elementary operations, which converges to the optimal value vector of the game. Implementations of those algorithms have been written and compared in [21].

Denote by n be the number of MAX vertices and r be the number of random vertices in an SSG. For SSGs with MAX vertices of outdegree 2, the best known deterministic algorithm is an SIA which makes at worst $O(2^n/n)$ iterations (see [25]), and the best known randomised algorithm is a SIA described by Ludwig in [22], which runs in $2^{O(\sqrt{n})}$.

Gimbert and Horn give an SIA in [16], running in $O^*(r!)$ iterations, namely a superpolynomial dependency in r only (O^* omits polynomial factors in r and n). For SSGs where random vertices have a probability distribution $(1/2, 1/2)$ (coin toss), Ibsen-Jensen and Miltersen present a VIA of complexity in $O^*(2^r)$ [18]. It turns out that all SIA runs in $O^*(2^r)$ on SSGs with probability distribution $(1/2, 1/2)$, as we prove in this article. The same complexity of $O^*(2^r)$ is obtained for general SSGs with a more involved randomised algorithm in [5].

Most of the aforementioned algorithms rely on the game being stopping, meaning that it structurally ends in a sink with probability 1. This condition is not restrictive since any SSG can be transformed into a stopping SSG while keeping the same optimal strategies. However, this transformation incurs a quadratic blow-up of the game and cannot be used in real life application. In this paper we give bounds in $O(2^r \text{Poly}(n))$ computational time for some kinds of algorithm. Using the stopping transformation would have induced a $O(2^{nr^2} \text{Poly}(n))$ complexity. The stopping restriction has been lifted for quadratic programming in [21] and before that for SIA and VIA in [10, 8].

Contributions

We introduce GSIA, a new meta-algorithm to solve SSGs in Sec. 3. This algorithm proves simultaneously the correctness of multiple algorithms ([14, 16, 15, 25, 18, 5]). In Sec. 4, we give a general complexity bound that matches or improves on previous bounds obtained by ad-hoc methods. We show that all these algorithms are fixed-parameter tractable in the number of random vertices. Moreover, we do not rely on the fact that the game is stopping, which was commonly used in the aforementioned papers. The proof of correctness relies on a notion of concatenation for strategies and an analysis of absorbing sets in the game, while the complexity bound is derived from a new and tight characterisation of the values of an SSG. Finally, in Sec. 5, we show how GSIA can be used to derive new algorithms, generalising classical ones. In particular, we exhibit a class of algorithms which generalise Gimbert and Horn's algorithm and use less iterations than Ibsen-Jensen and Miltersen's algorithm.

We emphasise that our goal here is not to define a new algorithm that would have a better –but still exponential– complexity bound than the state of the art (a sub-exponential algorithm for SSGs, like the ones found for parity games [6, 12], would already be a significant improvement), but rather to wrap-up a lot of previous research by showing that all known

SIA for SSGs, despite having emerged in different contexts and having ad-hoc proofs of convergence, are in fact instances of a general pattern that can be further expanded, and actually share the best known complexity bounds.

2 Simple Properties of Simple Stochastic Games

Some proofs are missing from this extended abstract for space reason, but they can be found in the appendices or in a long version [3].

2.1 Simple Stochastic Games

We give a generalised definition of *Simple Stochastic Game*, a two-player zero-sum game with turn-based moves and perfect information introduced by Anne Condon [14].

► **Definition 1.** *A Simple Stochastic Game (SSG) is a directed graph G , together with:*

1. *A partition of the vertex set V in four parts V_{MAX} , V_{MIN} , V_R and V_S (all possibly empty, except V_S), satisfying the following conditions:

 - a. *every vertex of V_{MAX} , V_{MIN} or V_R has at least one outgoing arc;*
 - b. *every vertex of V_S has exactly one outgoing arc which is a loop on itself.**
2. *For every $x \in V_R$, a probability distribution $p_x(\cdot)$ with rational values, on the outneighbourhood of x .*
3. *For every $x \in V_S$, a value $\text{Val}(x)$ which is a rational number in the closed interval $[0, 1]$.*

In the article, we denote $|V_{\text{MAX}}|$ by n and $|V_R|$ by r . Vertices from V_{MAX} , V_{MIN} , V_R and V_S are respectively called MAX vertices, MIN vertices, random vertices and sinks. For $x \in V$, we denote by $N^+(x)$ the set of outneighbours of x . We assume that for every $x \in V_R$ and $y \in V$, $y \in N^+(x)$ if and only if $p_x(y) > 0$.

The game is played as follows. The two players are named MAX and MIN. A token is positioned on a starting vertex x . If x is in V_{MAX} (resp. V_{MIN}) the MAX player (resp. the MIN player) chooses one of the outneighbours of x to move the token to. If x is in V_R , the token is randomly moved to one of the outneighbours of x according to the probability distribution $p_x(\cdot)$, independently of everything else. This process continues until the token reaches a sink s and then, player MIN has to pay $\text{Val}(s)$ to player MAX and the game stops. The problem we study is to *find the best possible strategies* for MIN and MAX, and the expected value that MIN has to pay to MAX while following those strategies.

We consider a slightly restricted class of SSGs where the probability distribution on each random vertex has a given precision and the value of the sinks are 0 and 1.

► **Definition 2.** *For q a positive integer, we say that an SSG is a q -SSG if there are only two sinks of value 0 and 1, and for all $x \in V_R$, there is an integer $q_x \leq q$ such that the probability distribution $p_x(\cdot)$ can be written as $p_x(x') = \frac{\ell_{x,x'}}{q_x}$ for all x' where $\ell_{x,x'}$ is a natural number.*

As an example, let x be a random vertex of a 2-SSG, and let $u \in N^+(x)$, then $p_x(u)$ can be equal to 0, $1/2$ or 1. The case $p_x(u) = 0$ is forbidden by definition, and if $p_x(u) = 1$, then x is of degree one and can be removed (by redirecting arcs entering x directly to u), without changing anything about the outcome of the game. Hence, we suppose without loss of generality that each random vertex of a 2-SSG has degree 2 and has probability distribution $(1/2, 1/2)$. This definition matches the one of a *binary SSG*, given by Condon and used in most articles on SSGs, except that we allow here MAX and MIN vertices to have an outdegree larger than 2.

2.2 Play, History and Strategies

► **Definition 3.** A **play** in G is an infinite sequence of vertices $X = (x_0, x_1, x_2, \dots)$ such that for all $t \geq 0$, (x_t, x_{t+1}) is an arc of G .

If for a play $X = (x_t)$ there is some $t \geq 0$ with $x_t = s \in V_S$, then all subsequent vertices in the play are also equal to s . In this case, we say that the play **reaches** sink vertex s and we define the **value of the play** $\text{Val}(X)$ as $\text{Val}(s)$. If the play reaches no sink, then we set $\text{Val}(X) = 0$.

A **history** of G is a finite directed path $h = (x_0, x_1, \dots, x_k)$. If the last vertex x_k is a MAX vertex (resp. MIN vertex), we say that h is a MAX history (resp. MIN history).

► **Definition 4.** A **general MAX strategy** (resp. **general MIN strategy**) is a map σ assigning to every MAX history (resp. MIN history) $h = (x_0, x_1, \dots, x_k)$ a vertex $\sigma(h)$ which is an outneighbour of x_k . The set of these strategies is denoted by Σ_{gen}^{MAX} (resp. Σ_{gen}^{MIN}).

For $\sigma \in \Sigma_{gen}^{MAX}$ and $\tau \in \Sigma_{gen}^{MIN}$, given a starting vertex x_0 , we recursively define a random play $X = (X_0, X_1, \dots)$ of G in the following way. At $t = 0$ let $X_0 = x_0$, and for $t \geq 0$:

- if $X_t \in V_{MAX}$, define $X_{t+1} = \sigma(X_0, X_1, \dots, X_t)$;
- if $X_t \in V_{MIN}$, define $X_{t+1} = \tau(X_0, X_1, \dots, X_t)$;
- if $X_t \in V_R$, then X_{t+1} is an outneighbour of X_t chosen following the probability distribution $p_{X_t}(\cdot)$, independently of everything else;
- if $X_t \in V_S$, define $X_{t+1} = X_t$.

This defines a distribution on plays which we denote by $\mathbb{P}_{\sigma, \tau}^{x_0}(\cdot)$, or simply $\mathbb{P}(\cdot)$ if strategies and starting vertex are clear from context. The corresponding expected value and conditional expected values are denoted by $\mathbb{E}_{\sigma, \tau}^{x_0}(\cdot|\cdot)$, or simply $\mathbb{E}(\cdot|\cdot)$.

We now define **positional strategies** which only depend on the last vertex in the history:

► **Definition 5.** A **general MAX strategy** σ (resp. **MIN strategy**) is said to be **positional** if for any MAX vertex x (resp. MIN vertex) and any history $h = (x_0, \dots, x)$, we have $\sigma(h) = \sigma((x))$ where (x) is the history containing only x as a start vertex. The set of positional MAX strategies (resp. MIN strategies) is denoted Σ^{MAX} (resp. Σ^{MIN}).

2.3 Values in an SSG

► **Definition 6.** Let G be an SSG and let (σ, τ) be a pair of MAX and MIN strategies, the **value vector** $v_{\sigma, \tau}^G$ is the real vector of dimension $|V|$ defined by, for any $x_0 \in V$,

$$v_{\sigma, \tau}^G(x_0) = \mathbb{E}_{\sigma, \tau}^{x_0}(\text{Val}(X)).$$

This value represents the expected gains for player MAX if both players plays according to (σ, τ) and the game starts in vertex x_0 .

As before, the superscript G can be omitted when the context is clear.

To compare value vectors, we use the pointwise order: we say that $v \geq v'$ if for all vertices $x \in V$ we have $v(x) \geq v'(x)$. Moreover, we say that $v > v'$ if $v \geq v'$ and there is some x such that $v(x) > v'(x)$. Given a MAX strategy σ , a **best response** to σ is a MIN strategy τ such that $v_{\sigma, \tau} \leq v_{\sigma, \tau'}$ for all MIN strategies τ' .

► **Proposition 7** ([14]). A positional strategy admits a positional best response, which can be found in polynomial time using linear programming.

The set of positional best responses to σ is denoted by $BR(\sigma)$. Similarly, for a MIN strategy τ , we define the notion of best response to τ and the corresponding set is denoted by $BR(\tau)$. Except explicitly stated otherwise (in Sec. 3.3), all considered strategies are positional.

We denote by $\tau(\sigma)$ a positional best response to σ . For a MAX strategy σ and $\tau \in BR(\sigma)$, we write v_σ for $v_{\sigma,\tau}$. For a MIN strategy τ and $\sigma \in BR(\tau)$, we write v_τ for $v_{\sigma,\tau}$. The vector v_σ is called the value vector of strategy σ , and is used to compare strategies by writing $\sigma' \underset{G}{>} \sigma$ if and only if $v_{\sigma'}^G > v_\sigma^G$.

It is well known (see [14, 25]) that there is a pair of deterministic positional strategies (σ^*, τ^*) called optimal strategies, that satisfies for all x , $v^* = v_{\sigma^*, \tau^*} = v_{\sigma^*} = v_{\tau^*}$ since σ^* and τ^* are best responses to each other.

2.4 Optimality Conditions

The next two lemmas give characterisations of (optimal) value vectors under a pair of strategies. They are fundamental to all algorithms finding optimal strategies. Proofs of similar results can be found in [13]; we add here a fifth condition to make the characterisation hold when the game is not stopping.

For any SSG, the vertices with value 0 under optimal strategies can be found in linear time by a simple graph traversal computing its complementary, the set of MAX vertices which can access a sink of positive value, regardless of the choice of the MIN player. Let K^G be the set of vertices with value 0 under optimal strategies. For a MAX strategy σ of G and a MIN strategy τ , we call $K_{\sigma,\tau}^G$ the set of vertices with value zero under the pair of strategies σ, τ , and K_σ^G the set of vertices with value zero under σ, τ when τ is a best response to σ .

► **Lemma 8.** *Given positional strategies (σ, τ) and a real $|V|$ -dimensional vector v , one has equality between v and $v_{\sigma,\tau}$ if and only if the following conditions are met:*

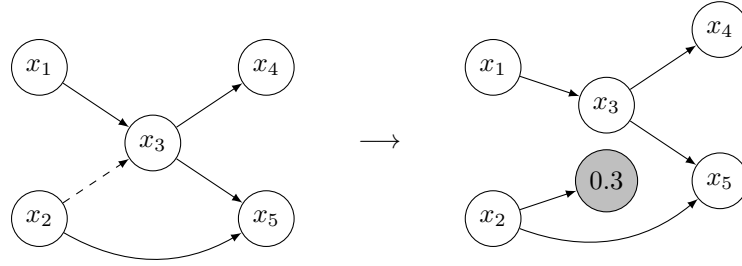
- (i) For $s \in V_S$, $v(s) = \text{Val}(s)$
- (ii) For $r \in V_R$, $v(r) = \sum_{y \in N^+(r)} p_r(y)v(y)$
- (iii) For $x \in V_{\text{MIN}}$, $v(x) = v(\tau(x))$
- (iv) For $x \in V_{\text{MAX}}$, $v(x) = v(\sigma(x))$
- (v) For any $x \in V$, $v(x) = 0$, if and only if $x \in K_{\sigma,\tau}^G$

Moreover, $\tau \in BR(\sigma)$ if and only if for any x in V_{MIN} , $v(x) = \min_{y \in N^+(x)} v(y) = v(\tau(x))$ and the last condition is modified into $v(x) = 0$ if and only if $x \in K_\sigma^G$.

► **Lemma 9 (Optimality conditions).** *Given positional strategies (σ, τ) and denoting $v = v_{\sigma,\tau}$, (σ, τ) are optimal strategies if and only if:*

- (i) For $s \in V_S$, $v(s) = \text{Val}(s)$
- (ii) For $r \in V_R$, $v(r) = \sum_{y \in N^+(r)} p_r(y)v(y)$
- (iii) For $x \in V_{\text{MIN}}$, $v(x) = \min_{y \in N^+(x)} v(y)$
- (iv) For $x \in V_{\text{MAX}}$, $v(x) = \max_{y \in N^+(x)} v(y)$
- (v) For any $x \in V$, $v(x) = 0$, if and only if $x \in K^G$

The conditions of Lemma 9 imply that (σ, τ) is a certificate of optimality that can be checked in polynomial time: compute $v_{\sigma,\tau}$ by solving the linear system of Lemma 8, compute K^G in linear time, then check in linear time if conditions are met.



■ **Figure 1** Transformation of the graph G in $G[\{(x_2, x_3)\}, f]$ where $f((x_2, x_3)) = 0.3$.

3 Generic Strategy Improvement Algorithm

3.1 Game Transformation

We present a simple transformation of an SSG, where some arcs of the game are rerouted to new sinks with appropriate values.

► **Definition 10.** Let G be an SSG, A be a subset of the arcs of G and f be a function from A to the set of rational numbers. Let $G[A, f]$ be the SSG obtained from a copy of G with the following modifications: each arc $e = (x, y) \in A$ is removed and replaced in $G[A, f]$ by $e' = (x, s_e)$ where s_e is a new sink vertex with value $f(e)$. These new sinks of $G[A, f]$ are called A -sinks, and A is called the set of fixed arcs.

Note that in the previous definition, the end vertex y of an arc $(x, y) \in A$ is not removed from the game. Its incoming arcs which are in A are simply redirected to sinks, see Fig. 1.

The function f is usually given by the values of a strategy: we denote by $G[A, \sigma]$ the game $G[A, f]$, where f is defined on every arc $e = (x, y)$ of A by $f(e) = v_\sigma(y)$. Comparing G and $G[A, \sigma]$, the only differences are that arcs of A have their endpoints changed to new sinks. Therefore, a strategy defined in G can be interpreted as a strategy of $G[A, \sigma]$ and vice versa, and we identify strategies in G and $G[A, \sigma]$. However, when we compare the values of a strategy in both games (as in Lemma 12 below), it makes sense to compare only the values on vertices in G and not on A -sinks (and anyway values of A -sinks are fixed).

► **Lemma 11.** For an SSG G , a subset of arcs A , and a MAX strategy σ , $K_\sigma^G = K_\sigma^{G[A, \sigma]}$.

Proof. Fix a min strategy τ and define $R_{\sigma, \tau}^G(x)$ as the set of vertices that can be reached from x in G , following only arcs corresponding to σ and τ after MAX and MIN vertices, and any arc out of random vertices. We repeatedly use the easy fact that the three following assertions are equivalent:

- (i) $v_{\sigma, \tau}^G(x) = 0$;
- (ii) $v_{\sigma, \tau}^G(y) = 0$ for all $y \in R_{\sigma, \tau}^G(x)$;
- (iii) $\text{Val}^G(s) = 0$ for all $s \in V_S^G \cap R_{\sigma, \tau}^G(x)$.

The same equivalence is true in $G[A, \sigma]$, where we define $R_{\sigma, \tau}^{G[A, \sigma]}$ likewise. Denote by $R_A^G(x)$ vertices of $R_{\sigma, \tau}^G(x)$ that are endpoints of arcs in A , and let $S_A(x)$ be the corresponding A -sinks in $G[A, \sigma]$.

Suppose that $v_{\sigma, \tau}^G(x) = 0$ and consider a sink s in $V_S^{G[A, \sigma]} \cap R^{G[A, \sigma]}(x)$: either it belongs to V_S^G hence also to $R^G(x)$ and satisfies $\text{Val}^G(s) = 0$ by (iii), or it belongs to $S_A(x)$ and then by definition

$$\text{Val}^{G[A, \sigma]}(s) = v_\sigma^G(s) \leq v_{\sigma, \tau}^G(s) = 0.$$

Thus, by (iii) once again we have $v_{\sigma, \tau}^{G[A, \sigma]}(x) = 0$.

Conversely, suppose that $v_{\sigma,\tau}^{G[A,\sigma]}(x) = 0$ and let $s \in V_S^G \cap R_{\sigma,\tau}^G(x)$. Then, either $s \in R_{\sigma,\tau}^{G[A,\sigma]}$, hence by (iii)

$$\text{Val}^G(s) = \text{Val}^{G[A,\sigma]}(s) = 0,$$

or there is a $y \in R_A^G(x)$ such that $s \in R_{\sigma,\tau}^G(y)$. In this case we have $v_{\sigma,\tau}^G(y) = 0$ by (ii), hence $\text{Val}^G(s) = 0$ by (iii) applied to y , and we see that $v_{\sigma,\tau}^G(x) = 0$.

Since we have $v_{\sigma,\tau}^G(x) = 0$ if and only if $v_{\sigma,\tau}^{G[A,\sigma]}(x) = 0$, regardless of τ , the result follows. \blacktriangleleft

► **Lemma 12.** For an SSG G , a subset of arcs A , and a MAX strategy σ , $v_\sigma^G = v_\sigma^{G[A,\sigma]}$.

Proof. This is a direct consequence of Lemma 8 and Lemma 11, since the vector v_σ^G satisfies the best-response conditions in $G[A,\sigma]$ and vice versa. \blacktriangleleft

3.2 The Algorithm

An SSG is **stopping** if under every pair of strategies, a play eventually reaches a sink with probability 1. Most algorithms in the literature depend on the game being stopping. It is usually not seen as a limitation since it is possible to transform every SSG into a stopping SSG, but the transformation makes the game polynomially larger by adding $O(nr)$ random vertices, which is bad from a complexity point of view, especially for algorithm with parametrized complexity in the number of random vertices. We strengthen the classical order on strategies, to get rid of the stopping condition in the generic strategy improvement algorithm presented in this section.

► **Definition 13.** Let σ and σ' be two MAX strategies, then $\sigma' \succ_G \sigma$ if $\sigma' \succ_G \sigma$ and for every MAX vertex x , if $v_{\sigma'}^G(x) = v_\sigma^G(x)$, then $\sigma'(x) = \sigma(x)$.

Algorithm 1 is a classical strategy improvement algorithm with two twists: the improvement is for *the stricter order* \succ and it is guaranteed in *the transformed game* rather than in the original game. We call Algorithm 1 the Generic Strategy Improvement Algorithm, or GSIA.

■ **Algorithm 1** GSIA.

Data: G a stopping SSG
Result: $(\sigma, \tau(\sigma))$ a pair of optimal strategies

```

1 begin
2   select an initial MAX strategy  $\sigma$ 
3   while  $(\sigma, \tau(\sigma))$  are not optimal strategies of  $G$  do
4     choose a subset  $A$  of arcs of  $G$ 
5     find  $\sigma'$  such that  $\sigma' \succ_{G[A,\sigma]} \sigma$ .
6      $\sigma \leftarrow \sigma'$ 
7   return  $(\sigma, \tau(\sigma))$ 

```

Algorithm 1 is a generic algorithm (or meta-algorithm) because neither the selection of an initial strategy σ at line 2, nor the way of choosing A at line 4, nor the way of finding σ' at line 5, are specified. A choice of implementation for these three parts is an **instance** of GSIA, that is a concrete strategy improvement algorithm.

Note that if $\sigma' \succ_{G[A,\sigma]} \sigma$ is found, it is easy to find σ'' with $\sigma'' \succ_{G[A,\sigma]} \sigma$: define σ'' as equal to σ' , except for MAX vertices x such that $v_{\sigma'}^G(x) = v_{\sigma}^G(x)$ and $\sigma'(x) \neq \sigma(x)$ where $\sigma''(x)$ is defined as $\sigma(x)$.

When we prove some property of GSIA in this article, it means that the property is true for all instances of GSIA, that is regardless of the selection of the initial strategy, the set A and the method for selecting σ' .

In order to prove the correctness of GSIA, we need to prove two points:

1. If σ is not optimal in G , then σ is not optimal in $G[A, \sigma]$.
2. If $\sigma' \succ_{G[A,\sigma]} \sigma$ then $\sigma' \succ_G \sigma$.

The first point is proved in the following lemma, while the second one is harder to obtain and is the subject of the next two subsections.

► **Lemma 14.** *For an SSG G and a subset of arcs A , a MAX strategy σ is optimal in G if and only if it is optimal in $G[A, \sigma]$.*

Proof. Except on A -sinks, the value vectors of σ in G and $G[A, \sigma]$ are equal by Lemma 12. Furthermore, by Lemma 11, $K_{\sigma}^G = K_{\sigma}^{G[A,\sigma]}$; hence σ satisfies the optimality conditions of Lemma 9 in G if and only if it satisfies them in $G[A, \sigma]$. ◀

3.3 Concatenation of Strategies

As a tool for proving the correctness of Algorithm 1, we introduce the notion of concatenation of strategies which produces non-positional strategies even if both concatenated strategies are positional. The idea of using a sequence of concatenated strategies to interpolate between two strategies has been introduced in [17].

► **Definition 15.** *For two MAX strategies σ, σ' and a subset of arcs A , we call $\sigma'|_A\sigma$ the non-positional strategy that plays like σ' until an arc of A is crossed, and then plays like σ until the end of the game. We let $\sigma'|_A^0\sigma = \sigma$ and for all $i \geq 0$, $\sigma'|_A^{i+1}\sigma = \sigma'|_A(\sigma'|_A^i\sigma)$.*

When A is clear from the context, we omit it and write $\sigma'|^i\sigma$. Strategy $\sigma'|_A^i\sigma$ is the strategy that plays like σ' until i arcs from A have been crossed, and then plays like σ . Hence, we can relate the strategy $\sigma'|_A\sigma$ to a positional strategy in $G[A, \sigma]$ as shown in the next lemma.

► **Lemma 16.** *For two MAX strategies σ, σ' and a subset of arcs A , we have: $v_{\sigma'|_A\sigma}^G = v_{\sigma'}^{G[A,\sigma]}$*

Proof. In G , after crossing an arc from A , by definition of $\sigma'|_A\sigma$, MAX plays according to σ . The game being memoryless, from this point, the best response for MIN is to play like $\tau(\sigma) \in BR(\sigma)$. Thus, there is a best response to $\sigma'|_A\sigma$ of the form $\tau'|\tau(\sigma)$ with τ' a MIN strategy not necessarily positional. Let us consider a play following $(\sigma'|_A\sigma, \tau|\tau(\sigma))$ with τ any MIN strategy. If the play does not cross an arc of A , then there is no difference between this play and a play following (σ', τ) in $G[A, \sigma]$. If an arc of A is used, then by Lemma 12 there is no difference between stopping with the value of $G[A, \sigma]$ or continuing in G while following (σ, τ) . Thus we have: $v_{\sigma'|_A\sigma, \tau|\tau(\sigma)}^G = v_{\sigma', \tau}^{G[A,\sigma]}$.

Thus, if τ' is a best response to σ' in $G[A, \sigma]$, then $\tau'|\tau(\sigma)$ is a best response to $\sigma'|_A\sigma$ in G . This implies that $v_{\sigma'|_A\sigma}^G = v_{\sigma'}^{G[A,\sigma]}$. ◀

We now prove the fact that increasing the values of sinks can only increase the value of the game (a similar lemma is proved in [4]).

► **Lemma 17.** *Let G and G' be two identical SSGs except the values of their sinks $s \in V_S$, denoted respectively by $Val(s)$ and $Val'(s)$. If for every $s \in V_S$, $Val'(s) \geq Val(s)$, then for every MAX strategy σ we have $v_{\sigma}^{G'} \geq v_{\sigma}^G$.*

Proof. For $s \in V_S$, let $\mathbb{P}_{\sigma,\tau}^x(\rightarrow s)$ be the probability that the play ends in sink s while starting from vertex x , following strategies (σ, τ) . For any vertex x we have:

$$v_{\sigma,\tau}^{G'}(x) = \sum_{s \in V_S'} \mathbb{P}_{\sigma,\tau}^x(\rightarrow s) Val'(s) \geq \sum_{s \in V_S'} \mathbb{P}_{\sigma,\tau}^x(\rightarrow s) Val(s) = v_{\sigma,\tau}^G(x)$$

This is true for any MIN strategy τ , thus $v_{\sigma}^{G'} \geq v_{\sigma}^G$. ◀

The following proposition is the core idea of GSIA: a strategy which improves on σ in the transformed game also improves on σ in the original game. The proof relies on a precise analysis of the set of vertices which cannot reach a sink, to deal with the fact that the game is not stopping. We prove that, if $\sigma' \succ_{G[A,\sigma]} \sigma$ the limit of $v_{\sigma'|^i \sigma}^G$ is $v_{\sigma'}^G$ and the two previous lemmas imply $\sigma'|^i \sigma \geq \sigma'|^{i-1} \sigma > \sigma$, which yields the following proposition.

► **Proposition 18.** *Let G be an SSG, A a subset of arcs of G and σ, σ' two MAX strategies. If $\sigma' \succ_{G[A,\sigma]} \sigma$ then $\sigma' \succ_G \sigma$.*

In order to avoid requiring the game to be stopping, it is necessary to pay particular attention to the set of vertices where the play can loop infinitely and yield value zero, which is a subset of the set of vertices of value 0. We now prove that a step of GSIA can only reduce this set, which is then used to prove Proposition 18.

► **Definition 19.** *For an SSG G and two strategies (σ, τ) , an absorbing set Z is a subset of $V \setminus V_S$ such that starting from any vertex of Z and playing according to (σ, τ) , there is a probability zero of reaching a vertex of $V \setminus Z$.*

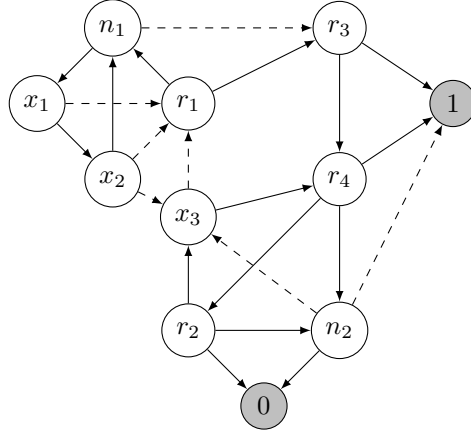
For σ and τ two strategies, $Z(\sigma, \tau)$ is the set of all vertices in some absorbing set under (σ, τ) . Hence, $Z(\sigma, \tau)$ is also an absorbing set. By definition, a play remains stuck in an absorbing set and can never reach a sink, hence all vertices of an absorbing set have *value zero* under (σ, τ) . The next lemma proves the existence of the inclusion-wise maximum over τ of $Z(\sigma, \tau)$ that we denote by $Z(\sigma)$. An example is given Fig. 2.

► **Lemma 20.** *For every MAX strategy σ , there is $\tau \in BR(\sigma)$ such that for every MIN strategy τ' , we have $Z(\sigma, \tau') \subseteq Z(\sigma, \tau)$.*

Proof. For τ in $BR(\sigma)$ and τ' such that $Z(\sigma, \tau') \not\subseteq Z(\sigma, \tau)$, then we define $\tilde{\tau}$ as $\tilde{\tau}(x) = \tau'(x)$ for x in $Z(\sigma, \tau')$ and $\tilde{\tau}(x) = \tau(x)$ otherwise. We now prove that $\tilde{\tau} \in BR(\sigma)$ and $Z(\sigma, \tilde{\tau}) \supseteq Z(\sigma, \tau) \cup Z(\sigma, \tau')$.

Since τ is a best response to σ , we have $v_{\sigma,\tau}(x) \leq v_{\sigma,\tau'}(x)$. Moreover, for $x \in Z(\sigma, \tau')$, $v_{\sigma,\tau'}(x) = 0$ thus $v_{\sigma,\tau}(x) = 0$. From this, we deduce that the two systems of linear equations given by Lemma 8, characterising respectively vectors $v_{\sigma,\tau}$ and $v_{\sigma,\tilde{\tau}}$, are exactly the same: for the only vertices where $\tilde{\tau}(x)$ and $\tau(x)$ differ satisfy $v_{\sigma,\tau}(\tau(x)) = v_{\sigma,\tau}(\tilde{\tau}(x)) = 0$. Hence, we have $v_{\sigma,\tau} = v_{\sigma,\tilde{\tau}}$ and $\tilde{\tau} \in BR(\sigma)$.

For any play under strategies $(\sigma, \tilde{\tau})$ starting in $x \in Z(\sigma, \tau')$, the MIN vertices of the play are all in $Z(\sigma, \tau')$ because $\tilde{\tau}$ plays as τ' on these vertices. Thus, we have $Z(\sigma, \tau') \subseteq Z(\sigma, \tilde{\tau})$. For a play starting in $x \in Z(\sigma, \tau)$, either the play reaches a vertex of $Z(\sigma, \tau')$ and then stays in $Z(\sigma, \tau')$ or it plays like τ and stays in $Z(\sigma, \tau)$. Hence, we have $Z(\sigma, \tau) \subseteq Z(\sigma, \tilde{\tau})$. ◀



■ **Figure 2** Example of an SSG where the x , n and r vertices are respectively from V_{MAX} , V_{MIN} and V_R . The pair of strategy (σ, τ) is displayed as plain arrows. Here $Z(\sigma, \tau) = Z(\sigma) = \{n_1, x_1, x_2\}$.

From this we deduce the following result on the improvement step for GSIA (where absorbing sets are understood in G):

► **Proposition 21** (Proof in Appendix A). *Let G be an SSG, A a set of arcs of G , σ and σ' two MAX strategies such that $\sigma' \succ_{G[A, \sigma]} \sigma$, then $Z(\sigma') \subseteq Z(\sigma)$.*

We now prove Proposition 18.

Proof. We introduce a sequence of non-positional strategies $(\sigma_i)_{i \geq 0}$ defined by $\sigma_i = \sigma'|^i \sigma$ for $i \geq 1$. By hypothesis $\sigma' \succ_{G[A, \sigma]} \sigma$, and by Lemma 16 $v_{\sigma'|_A \sigma}^G = v_{\sigma'}^{G[A, \sigma]}$, then we have

$$v_{\sigma_1}^G = v_{\sigma'|_A \sigma}^G = v_{\sigma'}^{G[A, \sigma]} > v_{\sigma}^{G[A, \sigma]} = v_{\sigma}^G.$$

Hence, by definition, sinks of $G[A, \sigma_1]$ will have at least the values of the corresponding sinks in $G[A, \sigma]$. Applying Lemma 17, we obtain that $v_{\sigma_1}^{G[A, \sigma_1]} \geq v_{\sigma}^{G[A, \sigma]}$, which can also be written as $v_{\sigma_2}^G \geq v_{\sigma_1}^G$. More generally, we have:

$$\forall i \geq 1, v_{\sigma_{i+1}}^G \geq v_{\sigma_i}^G \geq v_{\sigma_1}^G > v_{\sigma}^G.$$

We now prove that $v_{\sigma_i}^G \geq v_{\sigma_1}^G$ to conclude the proof.

From now on, we only consider the game G . Fix a vertex x and a MIN strategy $\tau \in BR(\sigma')$ such that $Z(\sigma') = Z(\sigma', \tau)$. From Proposition 21 we know that, $Z(\sigma') \subseteq Z(\sigma)$. It implies that for every $z \in Z(\sigma')$, $v_{\sigma}^G(z) = v_{\sigma'}^G(z) = 0$ which implies that $v_{\sigma'}^{G[A, \sigma]}(z) = 0$. Thus, $\sigma'(z) = \sigma(z)$. It implies that $Z(\sigma') \subseteq Z(\sigma, \tau)$.

We now only consider G' the game G where we replace every vertex in $Z(\sigma', \tau)$ by a sink of value 0. Lemma 12 directly implies that $v_{\sigma}^G = v_{\sigma'}^{G'}$ and $v_{\sigma'}^G = v_{\sigma'}^{G'}$. Moreover, when playing following σ_i when a vertex of $Z(\sigma')$ is reached, for all possible history, the play will stay in the absorbing set. Thus, $v_{\sigma_i}^G = v_{\sigma_i}^{G'}$.

Recall that $\mathbb{P}_{\sigma', \tau}^x(\rightarrow s)$ is the probability to reach a sink s in G' while starting in x and following (σ', τ) . Let $T^{\sigma', \tau}$ be a random variable defined as the time at which a sink is reached. Note that $T^{\sigma', \tau}$ may be equal to $+\infty$.

For every $i \geq 1$, we use Bayes rule to express the value of $v_{\sigma',\tau}(x)$ while conditioning on finishing the game before i steps.

$$\begin{aligned} v_{\sigma',\tau}(x) = & \mathbb{P}(T^{\sigma',\tau} < i) \sum_{s \in V_S} \mathbb{P}_{\sigma',\tau}^x(\rightarrow s \mid T^{\sigma',\tau} < i) \text{Val}(s) \\ & + \mathbb{P}(i \leq T^{\sigma',\tau} < +\infty) \sum_{s \in V_S} \mathbb{P}_{\sigma',\tau}^x(\rightarrow s \mid +\infty > T^{\sigma',\tau} \geq i) \text{Val}(s) \end{aligned}$$

If $T^{\sigma_i,\tau} < i$, only i arcs have been crossed, thus at most i arcs from A have been crossed when the sink is reached. Hence σ_i acts like σ' during the whole play, which yields:

$$\begin{aligned} v_{\sigma',\tau}(x) = & \mathbb{P}(T^{\sigma_i,\tau} < i) \sum_{s \in V_S} \mathbb{P}_{\sigma_i,\tau}^x(\rightarrow s \mid T^{\sigma_i,\tau} < i) \text{Val}(s) \\ & + \mathbb{P}(i \leq T^{\sigma_i,\tau} < +\infty) \sum_{s \in V_S} \mathbb{P}_{\sigma_i,\tau}^x(\rightarrow s \mid +\infty > T^{\sigma_i,\tau} \geq i) \text{Val}(s) \end{aligned}$$

We use Bayes rule in the same way for $v_{\sigma_i,\tau}(x)$

$$\begin{aligned} v_{\sigma_i,\tau}(x) = & \mathbb{P}(T^{\sigma_i,\tau} < i) \sum_{s \in V_S} \mathbb{P}_{\sigma_i,\tau}^x(\rightarrow s \mid T^{\sigma_i,\tau} < i) \text{Val}(s) \\ & + \mathbb{P}(i \leq T^{\sigma_i,\tau} < +\infty) \sum_{s \in V_S} \mathbb{P}_{\sigma_i,\tau}^x(\rightarrow s \mid T^{\sigma_i,\tau} \geq i) \text{Val}(s) \end{aligned}$$

Since every absorbing vertex in G associated with σ' has been turned into a sink, in G' $\mathbb{P}(T^{\sigma',\tau} < i) = \mathbb{P}(T^{\sigma_i,\tau} < i)$ converges to 1 when i grows. Hence, both $\mathbb{P}(i \leq T^{\sigma',\tau} < +\infty)$ and $\mathbb{P}(i \leq T^{\sigma_i,\tau} < +\infty)$ go to 0 and

$$\lim_{i \rightarrow +\infty} |v_{\sigma',\tau}(x) - v_{\sigma_i,\tau}(x)| = 0.$$

Hence, if there was x such that $v_{\sigma'}(x) < v_{\sigma_1}(x)$, we denote $\epsilon = v_{\sigma_1}(x) - v_{\sigma'}(x)$. For some rank I for all $i \geq I$ we have $|v_{\sigma',\tau} - v_{\sigma_i,\tau}| < \epsilon/2$. Which implies $v_{\sigma_i,\tau}(x) < v_{\sigma_1}(x)$. We recall that $v_{\sigma_1}(x) \leq v_{\sigma_i}(x)$. This means that $v_{\sigma_i,\tau} < v_{\sigma_i}(x)$, which contradicts the notion of optimal response against σ_i . Therefore, we have shown that $\sigma' \underset{G}{\geq} \sigma_1 \underset{G}{\geq} \sigma$. ◀

As a consequence of all previous lemmas, we obtain the correction of GSIA.

► **Theorem 22.** *GSIA terminates and returns a pair of optimal strategies.*

Proof. We denote by σ_i the MAX strategy σ at the end of the i -th loop in Algorithm 1. By induction, we prove that the sequence σ_i is of increasing value. Indeed, Line 5 of Algorithm 1 guarantees that $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$, thus Prop. 18 implies that $\sigma' \underset{G}{\succ} \sigma$, that is $\sigma_{i+1} > \sigma_i$.

The strategies produced by the algorithm are positional, hence there is only a finite number of them. Since the sequence is strictly increasing, it stops at some point. The algorithm only stops when Line 5 of Algorithm 1 fails to find $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$. In other words, σ is optimal in $G[A,\sigma]$. By Lemma 14, σ is also optimal in G . ◀

4 Complexity of GSIA

We analyse the algorithmic complexity of GSIA, by lower bounding the values of the sequence of strategies it produces. We obtain a bound on the number of iterations of GSIA depending on the number of random vertices, rather than on the number of MAX or MIN vertices. Then, we can derive the complexity of any instance of GSIA, by evaluating the cost of computing σ' from σ in $G[A, \sigma]$.

4.1 Values of q -SSGs

To prove a complexity bound using the values of a strategy, we need to precisely characterise the form of these values. In a 2-SSG, there is a function $f(r)$ such that, for every pair of positional strategies (σ, τ) , there is $t \leq f(r)$, such that for every vertex x , there is an integer p_x , such that $v_{\sigma, \tau}(x) = \frac{p_x}{t}$.

Condon proved in [13] that $f(r) \leq 4^r$. Then Auger, Coucheney and Strozecki improved this to $f(r) \leq 6^{r/2}$ in [4]. We show that $f(r) = q^r$ for q -SSGs, which gives the improved bound of $f(r) \leq 2^r$ for 2-SSGs.

► **Theorem 23** (Proof in Appendix B). *Let $q \geq 1$ and G a q -SSG with r random vertices, then for any pair of strategies (σ, τ) there is $t \leq q^r$ such that, for every vertex x , there is an integer s_x such that, $v_{\sigma, \tau} = \frac{s_x}{t}$.*

Proof of Th. 23 relies on the matrix tree theorem applied to a directed multigraph representing the game under a pair of strategies. Let us show that q^r is a tight bound for $f(r)$. Consider a Markov chain (an SSG with no MAX nor MIN vertices) with $r + 2$ vertices: two sinks 0 and 1 and r random vertices x_1, \dots, x_r . Vertex x_1 goes to 1 with probability $1/q$ and to 0 with probability $(q - 1)/q$. For $r \geq i \geq 2$, x_i goes to 0 with probability $(q - 1)/q$ and to x_{r-1} with probability $1/q$. Then, the value of x_r is q^{-r} .

4.2 Bounding the Number of Iterations of GSIA

GSIA produces a sequence of strictly increasing positional MAX strategies. The number of positional MAX strategies is bounded by $|\Sigma^{\text{MAX}}| = \prod_{x \in V_{\text{MAX}}} \text{deg}(x)$, hence the number of iterations of GSIA is bounded by this value. If we consider the case of a binary SSG (all vertices of outdegree 2), we have the classical bound of $|\Sigma^{\text{MAX}}| = 2^n$ iterations. The best known bound for a deterministic algorithm is $2^n/n$ iterations obtained for Hoffman-Karp algorithm [25], which is not far from the trivial bound of 2^n iterations.

We give a bound for q -SSG, which depends on q and r the number of random vertices. The difference of two values written as a/b and c/d , with a and b less than q^{-r} is more than q^{-2r} . Hence, if a value increases in GSIA, it increases at least by q^{-2r} . Using the classical notion of switch and anti-switch [25], recalled in Appendix C, we can prove that all vertices which have their value increased by a step of GSIA, are increased by at least q^{-r} .

► **Theorem 24** (Proof in Appendix C). *For G a q -SSG with r random vertices and n MAX vertices, the number of iterations of GSIA is at most nq^r .*

The complexity of GSIA is the number of iterations given by Th. 24, multiplied by the complexity of an iteration. In an iteration, there are two sources of complexity: constructing the game $G[A, \sigma]$ and finding an improving strategy σ' in $G[A, \sigma]$. To construct the game, v_σ is computed by solving a linear program of size m up to precision $p = q^r$. Let $C_1(m, p)$ be the

complexity of computing v_σ , then the best bound is currently in $O(m^\omega \log(p))$ [19], with ω the current best bound on the matrix multiplication exponent. Let $C_2(n, r, q)$ be the complexity of computing σ' , the complexity of an iteration is in $O(nq^r(C_1(n+r, q^r) + C_2(n, r, q)))$.

We obtain a better complexity, when $C_2(n, r, q) = O(C_1(n, q^r)r/n)$, which is the case for most instances of GSIA mentioned in this article. The number of iterations is only rq^r if we can guarantee that a random vertex increases its value at each step. When no random vertex is improved, the cost of computing $G[A, \sigma]$ can be made smaller, which yields the following theorem.

► **Theorem 25** (Proof in Appendix D). *Let G be a q -SSG with r random vertices and n MAX vertices. If $C_2(n, r, q) = O(C_1(n, q^r)r/n)$, then the complexity of GSIA is in $O(rq^r C_1(n, q^r))$.*

5 Two Instances of GSIA

As previously mentioned, all known strategy improvement algorithms can be viewed as particular instances of GSIA. This includes e.g. switch-based algorithms, like Hoffman-Karp algorithm [14, 25] or Ludwig's recursive algorithm [22]. With the help of GSIA it also becomes very easy to derive new algorithms, by transforming the game into polynomial time solvable instances, such as almost acyclic games [4]. We detail all these old and new algorithms in Sec. 6.

In this section, we focus instead on two particular instances (or family of instances) of GSIA, for which we obtain new complexity bounds using the results of the previous sections.

5.1 GSIA and f -strategies

The strategy improvement algorithm proposed by Gimbert and Horn in [16] (denoted by GHA) can be viewed as an instance of GSIA where the set A of fixed arcs is the set R of all arcs going out of random vertices, and the improvement step in the subgame $G[R, \sigma]$ consists in taking an optimal strategy. In this case, the subgame $G[R, \sigma]$ is deterministic (random vertices are connected to sinks only and can be replaced by sinks), hence optimal values in $G[R, \sigma]$ depend only on the relative ordering of the values $v_\sigma(x)$ for sink and random vertices x of G . These values can be computed in $O(r \log(r) + n)$ time [1]. In the original paper [16], the algorithm is proposed in a context where the number of sinks is two, but we generalise their definitions to our context.

Consider a total ordering f on $V_R \cup V_S$, $f : x_1 < x_2 < \dots < x_{r+s}$, where s is the number of sinks. An f -strategy corresponding to this ordering is an optimal MAX strategy in the game where the $s + r$ vertices above are replaced by sinks with new values satisfying $\text{Val}(x_1) < \text{Val}(x_2) < \dots < \text{Val}(x_{r+s})$. Clearly, this strategy does not depend on the actual values given but only on f . Note that if several f -strategies exist for a given f , they share the same values on all vertices.

Algorithm GHA produces an improving sequence of f -strategies, and the two sinks of value zero and one are always first and last in the order, hence its number of iterations is bounded by $r!$, the total number of possible orderings of the random vertices. We extend this result to a large class of instances of GSIA: let us call Optimal-GSIA (Opt-GSIA), the meta algorithm obtained from Algorithm 1 with two additional constraints:

- the set A of fixed arcs is the same at each step of Algorithm 1;
- at line 5, the improving strategy σ' is the optimal strategy in $G[A, \sigma]$.

All classical algorithms captured by GSIA, or new ones presented in this article are in fact instances of Opt-GSIA. We now show that Opt-GSIA has an iteration number similar to GHA. Since we have proved a bound of nq^r iterations, by Th. 24, Opt-GSIA has essentially the best known number of iterations, for q small and large (the latter being interesting in the case of random vertices with large degree and arbitrary probability distributions).

► **Theorem 26.** *Consider an SSG G and a set of arcs A containing k arcs out of MAX or MIN vertices. Then Algorithm Opt-GSIA runs in at most $\min((r+k)q^r, (r+k)!) iterations.$*

Proof. Let σ be one of the iterated MAX strategies obtained by an instance of Opt-GSIA, and σ' be an optimal strategy in $G[A, \sigma]$. Then σ' is consequently an f -strategy in $G[A, \sigma]$, where f is the ordering on $V_R \cup V_A$ (where V_A is the set of A -sinks) which is induced by the value vector $v_{\sigma'}^{G[A, \sigma]}$ (if vertices have the same value, just arbitrarily decide their relative ordering in f).

Since strategies produced by the algorithm strictly increase in values by Prop. 18, they must be all distinct. Hence, the order f must be distinct at each step of the algorithm, which proves that Opt-GSIA does at most $(r+k)!$ iterations.

Moreover, at every step the value in G of at least one vertex in $V_R \cup V_A$ must improve, by at least q^{-r} because of Th. 23. Since the value of these vertices is bounded by 1, the number of iterations of Opt-GSIA is bounded by $(r+k)q^r$. ◀

Those results can be used to generalize Gimbert and Horn's Algorithm [16] and to compare it with Ibsen-Jensen and Miltersen's algorithm [18]. It is possible to create an instance of GSIA with less iterations than Ibsen-Jensen and Miltersen's algorithm and which terminates exponentially faster on some input. Due to a lack of space this is detailed in a long version of this paper [3].

5.2 Condon's Converge From Below Algorithm

In [14], Condon first presents a faulty algorithm (the Naive Converge From Below Algorithm) and then a correct modified version, the Converge From Below (CFB) Algorithm. This algorithm proceeds by improving a value vector iteratively, but we show here that is in fact a disguised strategy improvement algorithm, that can be seen as an instance of Opt-GSIA. This gives us a proof of convergence of the CFB algorithm in the general, non-stopping case (whereas Condon has the assumption that the game is stopping in her proof), and also bounds on the number of iterations (none are given in the original paper) by Theorem 24 and Theorem 26.

The CFB algorithm is restated with some clarifications on listing 2 (we omit the details of the linear program, see [14]). The algorithm uses two properties of a vector, that we now define. First, vector v is *feasible* if

- (i) For $s \in V_S$, $v(s) = \text{Val}(s)$
- (ii) For $r \in V_R$, $v(r) = \sum_{x \in N_r} p_r(x)v(x)$
- (iii) For $x \in V_{\text{MIN}}$, $v(x) \leq \min_{y \in N^+(x)} v(y)$
- (iv) For $x \in V_{\text{MAX}}$, $v(x) \geq \max_{y \in N^+(x)} v(y)$.

A feasible vector is *stable* at x a MIN vertex (resp. MAX vertex) if satisfies condition (iii) (resp. condition (iv)) of feasibility for x with an equality.

We now show by induction that the CFB algorithm is equivalent to the instance of Opt-GSIA where all MIN vertices are fixed, i.e. A is the set of arcs entering MIN vertices. Let A_{MIN} denote this set.

To see this, suppose that at the beginning of line 5 of CFB, Vector v_r is the value vector of a MAX-strategy σ in G . Then:

- at Line 5, we “update v as the feasible vector where all MIN vertices x have value $v_r(x)$ and all MAX vertices are stable”. This amounts to finding a MAX-strategy σ' which satisfies optimality conditions in $G[A_{\text{MIN}}, \sigma]$, i.e. an optimal strategy for MAX in this subgame. This is exactly the subgame improvement step of Opt-GSIA. At the end of this step, v is the optimal value vector in $G[A_{\text{MIN}}, \sigma]$;
- in the next loop, at Line 4 of CFB, we “compute the value vector v_r of an optimal response to the MAX strategy that plays greedily according to v ”, i.e. v_r is updated to the value vector $v_{\sigma'}^G$. This is precisely Line 6 of GSIA when we update values in the subgame.

Hence, we see that except for the initialisation where v_r may not correspond to a MAX-strategy, it will be the case as soon as we reach Line 5 of the first loop, and from this point on CFB will correspond exactly to the instance of Opt-GSIA described above.

■ **Algorithm 2** Converge From Below Algorithm.

Data: G an SSG

Result: The optimal value vector v^* of G

```

1 begin
2   · let  $v$  be a feasible vector in which all MIN vertices have value 0 and all MAX
   vertices are stable
3   while  $v$  is not an optimal value vector do
4     · use linear programming to compute the value vector  $v_r$  of an optimal
     response to the MAX strategy that plays greedily according to  $v$ 
5     · update  $v$  as the feasible vector where all MIN vertices  $x$  have value  $v_r(x)$  and
     all MAX vertices are stable
6   return  $v$ 

```

6 Algorithms Derived from GSIA

We show that all known strategy improvement algorithms can be expressed as instances of GSIA and we also propose several new algorithms, derived from choices of A which make the transformed game polynomial time solvable. The only algorithms which are not instances of GSIA are based on values rather than strategies: value propagation [10, 14, 18], quadratic programming [14, 21] and dichotomy [4].

6.1 Hoffman-Karp Algorithms

The most classical method to solve an SSG, called the Hoffman-Karp algorithm, repeatedly applies switches to the strategy until finding the optimal one. It is also a generic algorithm, since the choice of the set of vertices to switch at each step is not specified nor the choice of the initial strategy. Many details on these algorithms can be found in [14] or [25].

Hoffman-Karp algorithms are instances of GSIA, where A is the set of all arcs of the SSG. Indeed, as proved in Lemma 30, a switch σ' of σ satisfies $\sigma \succ_{G[A, \sigma]} \sigma'$. Interpreting

Hoffman-Karp algorithms as instances of GSIA proves that they work on non-stopping games, while in most article the stopping condition is required. Moreover, it shows that their number of iterations is $O(nq^r)$ on q -SSGs, a complexity exponential in r only, which was known only for algorithms specially designed for this purpose [17, 15, 18, 5].

Ludwig’s Algorithm [22], which is the best randomised algorithm to solve SSGs, can be seen as an Hoffman-Karp algorithm using Bland’s rule as shown in [5]: a random order on the vertices is drawn, and at each step, the first switchable vertex in the order is switched. Two other Hoffman-Karp algorithms are presented in [25]: switching all switchable vertices at each step or switching a random subset. Seeing these three algorithms as instances of GSIA yields $O(nq^r)$ as a deterministic bound on their number of iterations, which was unknown. However, the analysis of [22, 25] is required to obtain a good complexity in n for these algorithms.

6.2 Selection of the Initial Strategy

In [15], Dai and Ge give a randomised improvement of GHA simply by choosing a better initial strategy. To do so, they choose randomly $\sqrt{r!} \log(r!)$ strategies and choose the one with the highest value. This ensures, with high probability, that at most $\sqrt{r!}$ iterations will be done in GHA. Thus, their algorithm runs in $O(\sqrt{r!})$ iterations. This algorithm is also captured by GSIA by selecting the initial strategy in the same way, however it seems hard to combine the gain made by the random selection of the strategy and the bound in $O(q^r)$ of GSIA, since even a strategy close to the optimal one may have values far from it. Remark that it is trivial to extend this method to any instance of Opt-GSIA to improve on the complexity of Th. 26.

6.3 New Algorithms

We can use GSIA to design many strategy improvement algorithms. We present three of them, all based on a choice of A which makes $G[A, \sigma]$ solvable in polynomial time. The initial strategy can be anything and σ' is always chosen to be the optimal strategy in $G[A, \sigma]$. Most of them can be seen as generalisations of known algorithms.

1. Let A be a feedback arc set of G , then $G[A, \sigma]$ is acyclic and it can be solved in linear time. It seems intuitively appealing to think that this algorithm will be faster if the feedback arc set is small but we have no proof to sustain such a proposition.
2. A MAX acyclic SSG is an SSG such that every MAX vertex has at most one outgoing arc in a cycle. MAX acyclic SSG can be solved in polynomial time, see [4]. If we let A be a set of arc that contains all but one outgoing arcs of each MAX vertex, then $G[A, \sigma]$ is MAX acyclic and can be solved in polynomial time. Moreover, such a game can be solved by strategy improvement in at most n iterations. This can be seen as a generalisation of Hoffman-Karp algorithm, in which A contains *all* outgoing arcs of MAX vertices.
3. As an intermediate between acyclic games and MAX acyclic games, we may consider almost acyclic games, where all vertices have at most one outgoing arc in a cycle. Almost acyclic SSGs can be solved in linear time [4].

References

- 1 Daniel Andersson, Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. Deterministic graphical games revisited. In *Conference on Computability in Europe*, pages 1–10. Springer, 2008.
- 2 Daniel Andersson and Peter Bro Miltersen. The complexity of solving stochastic games on graphs. In *International Symposium on Algorithms and Computation*, pages 112–121, 2009.
- 3 David Auger, Xavier Badin de Montjoye, and Yann Strozecki. A generic strategy iteration method for simple stochastic games. *arXiv preprint*, 2021. [arXiv:2102.04922](https://arxiv.org/abs/2102.04922).

- 4 David Auger, Pierre Coucheney, and Yann Strozecki. Finding optimal strategies of almost acyclic simple stochastic games. In *International Conference on Theory and Applications of Models of Computation*, pages 67–85, 2014.
- 5 David Auger, Pierre Coucheney, and Yann Strozecki. Solving Simple Stochastic Games with Few Random Nodes Faster Using Bland’s Rule. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, pages 9:1–9:16, 2019.
- 6 Cristian S Calude, Sanjay Jain, Bakhadyr Khossainov, Wei Li, and Frank Stephan. Deciding parity games in quasi-polynomial time. *SIAM Journal on Computing*, 0(0):STOC17–152, 2020.
- 7 Seth Chaiken and Daniel J Kleitman. Matrix tree theorems. *Journal of combinatorial theory, Series A*, 24(3):377–381, 1978.
- 8 Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Strategy improvement for concurrent reachability and turn-based stochastic safety games. *Journal of Computer and System Sciences*, 79(5):640–657, 2013.
- 9 Krishnendu Chatterjee and Nathanaël Fijalkow. A reduction from parity games to simple stochastic games. *Electronic Proceedings in Theoretical Computer Science*, 54, 2011.
- 10 Krishnendu Chatterjee and Thomas A. Henzinger. *Value Iteration*, pages 107–138. Springer Berlin Heidelberg, 2008.
- 11 Taolue Chen, Marta Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In *Quantitative Evaluation of Systems*, pages 322–337, 2013.
- 12 Thomas Colcombet and Nathanaël Fijalkow. Universal graphs and good for games automata: New tools for infinite duration games. In *International Conference on Foundations of Software Science and Computation Structures*, pages 1–26. Springer, 2019.
- 13 Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.
- 14 Anne Condon. On algorithms for simple stochastic games. *Advances in computational complexity theory*, 13:51–73, 1993.
- 15 Decheng Dai and Rong Ge. New results on simple stochastic games. In *International Symposium on Algorithms and Computation*, pages 1014–1023. Springer, 2009.
- 16 Hugo Gimbert and Florian Horn. Simple stochastic games with few random vertices are easy to solve. In *Foundations of Software Science and Computational Structures*, pages 5–19. Springer, 2008.
- 17 Hugo Gimbert and Florian Horn. Solving simple stochastic games with few random vertices. *Logical Methods in Computer Science*, Volume 5, Issue 2, 2009.
- 18 Rasmus Ibsen-Jensen and Peter Bro Miltersen. Solving simple stochastic games with few coin toss positions. In *European Symposium on Algorithms*, pages 636–647. Springer, 2012.
- 19 Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. *arXiv preprint*, 2020. [arXiv:2004.07470](https://arxiv.org/abs/2004.07470).
- 20 Brendan Juba. On the hardness of simple stochastic games. *Master’s thesis, CMU*, 2005.
- 21 Jan Křetínský, Emanuel Ramneantu, Alexander Slivinskiy, and Maximilian Weininger. Comparison of algorithms for simple stochastic games. *arXiv preprint*, 2020. [arXiv:2009.10882](https://arxiv.org/abs/2009.10882).
- 22 Walter Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and computation*, 117(1):151–155, 1995.
- 23 L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.
- 24 C Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of the IGPL*, 7(1):103–124, 1999.
- 25 Rahul Tripathi, Elena Valkanova, and VS Anil Kumar. On strategy improvement algorithms for simple stochastic games. *Journal of Discrete Algorithms*, 9(3):263–278, 2011.

A Proof of the decreasing absorbing set

Here we prove Prop. 21

Proof. Suppose that $Z(\sigma')$ is not a subset of $Z(\sigma)$. From Lemma 20 there is $\tau \in BR(\sigma)$ such that $Z(\sigma, \tau) = Z(\sigma)$ and $\tau' \in BR(\sigma')$ such that $Z(\sigma', \tau') = Z(\sigma')$. We write $Z = Z(\sigma')$

Let X be the set of MAX vertices x in $Z(\sigma') \setminus Z(\sigma)$ such that $\sigma(x) \neq \sigma'(x)$; it is nonempty otherwise $Z(\sigma')$ would be an absorbing set for (σ, τ') . If x is in X , since $\sigma' \succ_{G[A, \sigma]} \sigma$, we have

$$v_{\sigma'}^{G[A, \sigma]}(x) > v_{\sigma}^{G[A, \sigma]}(x) \geq 0$$

Thus, a sink is reached in $G[A, \sigma]$ starting from x under the strategies (σ', τ') . Since Z is an absorbing set in G under the same strategies, it implies that all the accessible sinks in $G[A, \sigma]$ are A -sinks. Hence, there is at least one arc $e = (y, z) \in A$ with both ends in Z and such that $v_{\sigma}(z) > 0$. We define the vertex s of Z as:

$$s = \arg \max_{z \in Z} \{v_{\sigma}(z) \mid \exists y \in V, (y, z) \in A\}$$

and we let $v = v_{\sigma}(s)$. The value of each vertex in Z is bounded by v . Similarly than for x , in G under strategies (σ, τ) the value of s is bounded by the value of the vertex leaving Z . Such vertices exist since Z is not a subset of $Z(\sigma)$. We now want to show that those vertices all have value strictly lesser than v , thus proving a contradiction.

First, since Z is an absorbing set for (σ', τ') , all arcs leaving a random vertex in $Z(\sigma')$ remain in $Z(\sigma')$ in G ; this is not dependent on the strategies considered.

Let $E_X \subseteq X$ the set of MAX vertices x of X such that $\sigma(x) \notin Z$ and let $E_N \subseteq Z \cap V_{\text{MIN}}$ the set of MIN vertices x of Z such that $\tau(x) \notin Z$.

On the one hand, for a MIN vertex $x \in E_N$:

$$\begin{aligned} v_{\sigma}^G(\tau(x)) &\leq v_{\sigma}^G(\tau'(x)) && \text{Since } \tau = \tau(\sigma) \\ v_{\sigma}^G(\tau'(x)) &= v_{\sigma}^{G[A, \sigma]}(\tau'(x)) \\ v_{\sigma}^{G[A, \sigma]}(\tau'(x)) &\leq v_{\sigma'}^{G[A, \sigma]}(\tau'(x)) && \text{Since } \sigma' \succ_{G[A, \sigma]} \sigma \\ v_{\sigma'}^{G[A, \sigma]}(\tau'(x)) &\leq v && \text{Since } \tau'(x) \in Z \end{aligned}$$

Thus, $v_{\sigma}^G(\tau(x)) \leq v$. In case of equality, we have $v = v_{\sigma}^G(\tau'(x)) = v_{\sigma}^G(\tau(x))$; hence we can replace τ by $\bar{\tau}(x)$, which is identical to τ except that $\bar{\tau}(x) = \tau'(x)$. We have $v_{\sigma, \tau} = v_{\sigma, \bar{\tau}}$ and $Z(\sigma, \tau) = Z(\sigma, \bar{\tau})$. Indeed, according to Lemma 8 the only situation that could occur would be to violate the condition (v) by creating an absorbing set. However this would contradict the definition of τ . Thus, we can suppose that for any x in E_N , $v_{\sigma}(\tau(x)) < v$.

On the other hand, since $\sigma' \succ_{G[A, \sigma]} \sigma$ we know that for any x in E_X :

$$v_{\sigma}^G(\sigma(x)) < v_{\sigma'}^{G[A, \sigma]}(x) \leq v$$

Now, for any vertex x of $E = E_X \cup E_N$, let p_x be the probability of x being the first vertex of E reached starting from s following strategies (σ, τ) . By conditional expectation:

$$v_{\sigma}(s) = \sum_{x \in E_X} p_x v_{\sigma}(\sigma(x)) + \sum_{x \in E_N} p_x v_{\sigma}(\tau(x))$$

Thus, $v_{\sigma}(s) < v$ which contradicts the definition of v , and proves that $Z(\sigma') \subseteq Z(\sigma)$. ◀

B Proof of the value of a q -SSG

Here, we prove Th. 23.

Let us remark that a q -SSG can be assumed to have all its probability transition of the form p/q . The idea here is to notice that it is possible to loop with a certain probability on the same random vertices.

► **Lemma 27.** *Let G be a q -SSG, then there is G' a q -SSG with the same vertices which defines the same expectation $\mathbb{E}_{\sigma, \tau}^{x_0}(\cdot)$ and such that for all $x \in V_R$ and all $x' \in N^+(x)$ then there is an integer $p_{x,x'}$ such that $p_x(x') = p_{x,x'}/q$.*

Proof. For a a random vertex in G , and $q_a < q$ such that for every other vertex x in G there is $p_x \in \mathbb{N}$ and a probability p_x/q_a to go directly from a to x , we change those probabilities to p_x/q and we add a probability p/q to stay in a , where:

$$p = q - \sum_{x \in V} p_x. \quad \blacktriangleleft$$

Now, we state the classical matrix-tree theorem that we use in our proof (see e.g. [7]). Let G be a directed multigraph with n vertices, then the *Laplacian matrix* of \mathcal{G} is a $n \times n$ matrix $L(\mathcal{G}) = (l_{i,j})_{i,j \leq n}$ defined by:

- (i) $l_{i,j}$ equals $-m$ where m is the number of arcs from i to j .
- (ii) $l_{i,i}$ is the number of arcs going to i , excluding the self-loops.

► **Theorem 28** (Matrix tree theorem for directed multigraphs). *For $G = (V, E)$ a directed multigraph with vertices $V = \{v_1, \dots, v_k\}$ and L its Laplacian matrix, the number of spanning trees rooted at v_i is $\det(\hat{L}_{i,i})$ where $\hat{L}_{i,i}$ is the matrix obtained by deleting the i -th row and column from L .*

We can now prove Th. 23.

Proof of Th. 23. The beginning of the proof is the same as in [14] and [4]. We start by transforming the game with fixed strategies in a Markov Chain with equivalent values. Then, we show that the value of each vertex can be written $\frac{\det B_i}{\det q(I - A)}$ using Cramer rule, for B_i and A two matrix which will be carefully defined. To conclude, we will show that $\det q(I - A) < q^r$ by creating a graph obtain from our initial game and using Th. 28.

We consider a q -SSG G and two positional strategies σ and τ . Without loss of generality, we can restrict ourselves to the computation of non-zero, non-sink values. Thus, each vertex has a non-zero probability to reach the 1-sink. To compute the values $v_{\sigma, \tau}$, we can consider G_A an SSG with vertices $V_R \cup V_S$: the random vertices and the sinks of V . The value of the sinks is not changed and the probability distribution p'_x is defined as follows. For $x \in V_R$ and x' in G_A , we call $M_{x,x'}$ the set of MAX and MIN vertex y in $N^+(x)$ such that there is a path following only arcs of σ and τ from y to x' . We then have

$$p'_x(x') = \sum_{y \in M_{x,x'}} p_x(y)$$

The graph G_A has $r + 2$ vertices that we denote by $a_1, \dots, a_{r+1}, a_{r+2}$ where a_{r+1} is the 0-sink and a_{r+2} is the 1-sink. Let b be the r -dimensional column vector with $b_i = p'_{a_i}(a_{r+2})$. We define A the $r \times r$ matrix, with $A_{i,j} = p'_{a_i}(a_j)$.

12:20 A Generic Strategy Improvement Method for Simple Stochastic Games

The values of the random vertices are defined by the vector z that satisfies the following equation:

$$z = Az + b$$

Let I be the identity matrix, $(I - A)$ is invertible because each random vertex has access to a sink and every eigenvalue of A is strictly less than 1. We refer to [14] for details. Hence, the equation has a unique solution and z is also solution of:

$$q(I - A)z = qb$$

Hence, under the strategies σ, τ , the value z_i of a random vertex a_i given by the Cramer rule is

$$z_i = \frac{\det B_i}{\det q(I - A)}$$

where B_i is the matrix $q(I - A)$ where the i -th column has been replaced by qb . The value $\det B_i$ is an integer. See [4] for more details. Our goal is now to bound $\det q(I - A)$.

From the graph G_A , we construct the graph G' by inverting all arcs, and duplicating an arc of probability p/q into p arcs of probability $1/q$. We also add an arc coming from the 1-sink to the 0-sink and one from the 0-sink towards the 1-sink. Figure 3 shows an example of the transformation from G to G' . The Laplacian L of G' is thus the following matrix.

$$L = \left(\begin{array}{c|cc} q(I - A)^T & B & \\ \hline 0 & 1 & 1 \\ & 1 & 1 \end{array} \right)$$

Indeed, every random vertex has indegree q minus the number of loops. Thus the number of spanning trees of G' rooted in the 1-sink is equal by Th. 28 to $\det \hat{L}_{r+2, r+2}$ where we have

$$\hat{L}_{r+2, r+2} = \left(\begin{array}{c|c} q(I - A)^T & B' \\ \hline 0 & 1 \end{array} \right).$$

In other words, the number of spanning trees of G' is equal to $\det q(I - A)$. Furthermore, each spanning tree contains exactly one incoming arcs for every random vertices, and the arc (a_{r+2}, a_{r+1}) has to be used. Thus, there is at most q^r spanning trees rooted in G' and $\det q(I - A) \leq q^r$. ◀

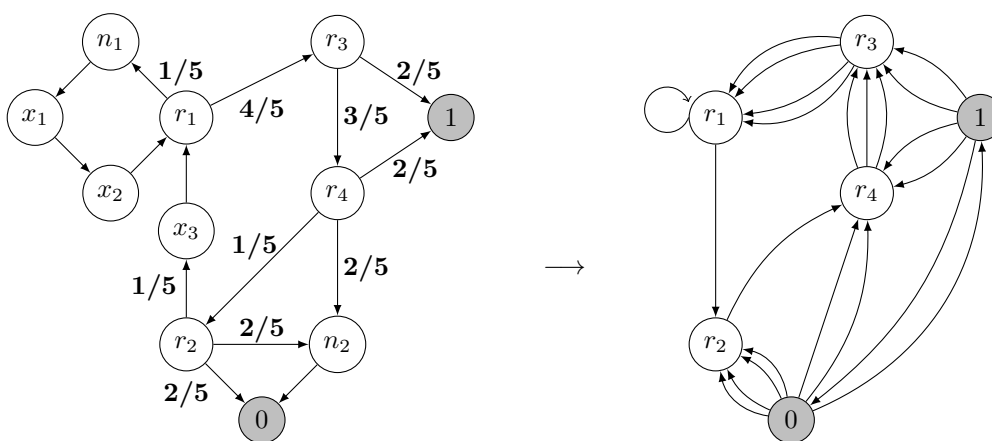
C Bounding the Number of Iterations of GSIA

We introduce the notion of switch and anti-switch, to prove that the improvement is at least q^{-r} rather than q^{-2r} .

► **Definition 29.** A *switch* (resp. an *anti-switch*) of a MAX strategy σ with switched set $S \subseteq V_{\text{MAX}}$ is a strategy σ_S defined by $\sigma_S(x) = \sigma(x)$ for $x \notin S$, and satisfying $v_\sigma(\sigma(x)) < v_\sigma(\sigma_S(x))$ (resp. $v_\sigma(\sigma(x)) > v_\sigma(\sigma_S(x))$) for $x \in S$ (hence $\sigma_S(x) \neq \sigma(x)$).

A common tool to solve SSGs is the fact that a switch increases the value of a strategy, while an anti-switch decreases it. Within our framework of transformed game, it is extremely simple to prove.

► **Lemma 30.** If σ_S is a switch of σ , then $\sigma_S > \sigma$. If σ_S is an anti-switch of σ , then $\sigma_S < \sigma$.



■ **Figure 3** Example of a transformation of a graph G into a graph G' .

Proof. Consider $G[A, \sigma]$ the game obtained from G , where A is the set of all arcs of G . Let us consider x a vertex switched in σ' , that is with $v_\sigma^G(\sigma(x)) < v_{\sigma'}^G(\sigma'(x))$. Then, because all arcs are in A , we have $v_\sigma^{G[A, \sigma]}(x) = v_\sigma^G(\sigma(x))$ and $v_{\sigma'}^{G[A, \sigma]}(x) = v_{\sigma'}^G(\sigma'(x))$. Hence, $v_\sigma^{G[A, \sigma]}(x) < v_{\sigma'}^{G[A, \sigma]}(x)$ and for $v_\sigma^G(\sigma(x)) \geq v_{\sigma'}^G(\sigma'(x))$, $\sigma(x) = \sigma'(x)$, which implies $\sigma' \succ_{G[A, \sigma]} \sigma$.

Prop. 18 proves $\sigma' \succ_G \sigma$.

The proof is the same for an anti-switch, since $\sigma \succ_{G[A, \sigma]} \sigma' \Rightarrow \sigma \succ_G \sigma'$ (which can be proved similarly as Prop. 18, while keeping in mind that in the decreasing case, creating absorbing set lowers the value). ◀

We use the previous lemma to prove Th. 24.

Proof. Let us consider σ the strategy computed at some point by GSIA and σ' the next strategy. By Prop. 18, $\sigma < \sigma'$. Hence, by Lemma 30, σ' cannot be an anti-switch of σ . Thus, there is a MAX vertex x such that $v_\sigma(\sigma(x)) < v_{\sigma'}(\sigma'(x))$. We recall that $\sigma'(x)$ denotes the successor of x under strategy σ' .

Since $\sigma < \sigma'$, we have $v_\sigma(x) = v_\sigma(\sigma(x)) < v_\sigma(\sigma'(x)) \leq v_{\sigma'}(\sigma'(x)) = v_{\sigma'}(x)$. We now evaluate $v_{\sigma'}(\sigma'(x)) - v_\sigma(\sigma(x))$. In the game G , under the strategies $\sigma, \tau(\sigma)$, Th. 23 implies that for some $t \leq q^r$, $v_\sigma(\sigma(x)) = p/t$ and $v_\sigma(\sigma'(x)) = p'/t$. We have $p/t < p'/t$, thus $p'/t - p/t \geq 1/t \geq 1/q^r$. Hence, the value of some MAX vertex increases by $1/q^r$ in each iteration of GSIA. Since there are n MAX vertices and their values are bounded by 1, there are at most nq^r iterations. ◀

D Amortised Complexity of GSIA

Here, we prove Th. 25.

Proof. We assume that $r < n$, otherwise the theorem is trivial. Let σ' be the strategy computed by GSIA at some point, improving on the strategy σ . GSIA must compute $G[A, \sigma']$, and thus $v_{\sigma'}$ and we explain a method to do so efficiently.

We assume that the order of the values (in G) of the random vertices is the same for σ and σ' . Then, knowing this order and σ' , it is easy to compute $\tau(\sigma')$ a best response to σ' in $O(r \log(r) + n)$ time [1]. Then, we can compute the values $v_{\sigma', \tau(\sigma')}$ in time $O(C_1(r, q^r))$, since it is done by solving a linear system of dimension r with precision q^r , a task which

12:22 A Generic Strategy Improvement Method for Simple Stochastic Games

is simpler than solving a linear program. Since $C_1(r, q^r)$ is at least quadratic in r , then $C_1(r, q^r) < C_1(n, q^r)r/n$ and by hypothesis $C_2(n, r, q) = O(C_1(n, q^r)r/n)$, hence a step is of complexity at most $O(C_1(n, q^r)r/n)$. There are at most nq^r such steps, for a total complexity of $O(rq^r C_1(n, q^r))$.

We need to detect when the assumption that the values of the random vertices are the same for σ and σ' is false. If $v_{\sigma', \tau(\sigma')}$ satisfies the optimality conditions at the MIN vertices, then $\tau(\sigma')$ is a best response. Otherwise, we compute the best response by solving a linear program in time $C(n, q^r)$. In that case, the order of the random vertices has changed: there are two vertices x_1 and x_2 such that $v_\sigma(x_1) < v_\sigma(x_2)$ and $v_{\sigma'}(x_1) > v_{\sigma'}(x_2)$. Hence, $v_{\sigma'}(x_1) > v_\sigma(x_2)$, which implies that $v_{\sigma'}(x_1) - v_\sigma(x_1) > v_\sigma(x_2) - v_\sigma(x_1) > q^{-r}$.

We have proved that when the random order changes, the value of some random vertex increases by at least q^{-r} , hence there are at most nq^r such steps. The complexity from these steps is bounded by $O(rq^r C_1(n, q^r))$, which proves the theorem. ◀

(Un)Decidability for History Preserving True Concurrent Logics

Paolo Baldan  

University of Padova, Italy

Alberto Carraro  

ITIS Zuccante, Venezia, Italy

Tommaso Padoan  

University of Padova, Italy

Abstract

We investigate the satisfiability problem for a logic for true concurrency, whose formulae predicate about events in computations and their causal (in)dependencies. Variants of such logics have been studied, with different expressiveness, corresponding to a number of true concurrent behavioural equivalences. Here we focus on a mu-calculus style logic that represents the counterpart of history-preserving (hp-)bisimilarity, a typical equivalence in the true concurrent spectrum of bisimilarities.

It is known that one can decide whether or not two 1-safe Petri nets (and in general finite asynchronous transition systems) are hp-bisimilar. Moreover, for the logic that captures hp-bisimilarity the model-checking problem is decidable with respect to prime event structures satisfying suitable regularity conditions. To the best of our knowledge, the problem of satisfiability has been scarcely investigated in the realm of true concurrent logics.

We show that satisfiability for the logic for hp-bisimilarity is undecidable via a reduction from domino tilings. The fragment of the logic without fixpoints, instead, turns out to be decidable. We consider these results a first step towards a more complete investigation of the satisfiability problem for true concurrent logics, which we believe to have notable solvable cases.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Modal and temporal logics

Keywords and phrases Event structures, history-preserving bisimilarity, true concurrent behavioural logics, satisfiability, decidability, domino systems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.13

Funding This work is supported by the Ministero dell'Università e della Ricerca Scientifica of Italy, under Grant No. 201784YSZ5, PRIN2017 – ASPRA.

1 Introduction

When dealing with concurrent and distributed systems, the so-called true concurrent models are used to provide a precise account of the computational steps and of their dependencies, like causality and concurrency. An early and widely used foundational model in this class is given by Winskel's event structures [41]. They describe the behaviour of a system in terms of events in computations and two dependency relations: a partial order modelling causality and an additional relation representing conflict. A survey on the applications of such causal models can be found in [42]. Recently they have been used in the study of concurrency in weak memory models [31, 20], for process mining and differencing [14], in the study of atomicity [15] and information flow [2] properties.

In the true concurrent approach numerous behavioural equivalences have been defined ranging from hereditary history-preserving bisimilarity to the coarser pomset and step equivalences (see, e.g., [37]). Correspondingly, behavioural logics have been proposed including operators that allow one to express causal properties of computations (see, e.g., [12, 7, 32, 29, 25, 11, 33] just to mention a few).



© Paolo Baldan, Alberto Carraro, and Tommaso Padoan;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 13; pp. 13:1–13:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In particular, event-based logics have been introduced [3, 30], interpreted over event structures, expressive enough to provide a logical characterization of the main behavioural equivalences in the true concurrent spectrum [37], from hereditary history-preserving (hhp-)bisimilarity [7] to the coarser equivalences. Formulae of such logics include variables which can be bound to events in computations and describe their dependencies.

The relation between operational models, behavioural equivalences, and true concurrent logics has been widely studied and the model-checking problem has been investigated for various logics describing true concurrency properties (see, e.g., [1, 18, 16, 17, 23, 6, 27]). The decidability of true concurrent equivalences has also been settled in various papers.

A natural problem that, to the best of our knowledge, has been scarcely investigated for true concurrent logics is satisfiability, which has been historically referred to as the *classical decision problem* [10] in the context of first-order logic. The satisfiability problem for true concurrent logics is the quest for an algorithm that, given as input any formula φ , determines whether or not there exists an event structure that satisfies φ . From this point of view, formulae are intended as abstract specifications of desired properties and event structures are abstractions of actual systems that, if implemented, should have those properties. An algorithm for satisfiability is therefore a sort of oracle that can tell system designers whether or not their desires can be realized. Obviously, checking satisfiability allows one also to verify whether two requirements, despite being syntactically different, are equivalent.

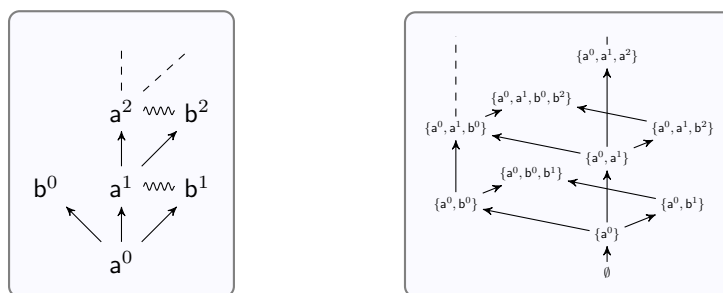
In this paper we tackle the satisfiability problem for the logic proposed in [3], referred to as \mathcal{L}_{hp} , corresponding to history-preserving (hp-)bisimilarity [9, 34, 13], a classical equivalence in the spectrum. The logic is endowed with least and greatest fixpoint operators, in mu-calculus style, in order to express interesting properties of infinite computations. For the propositional mu-calculus, corresponding to ordinary interleaving bisimilarity, satisfiability is decidable and every satisfiable formula has a finite model. For \mathcal{L}_{hp} instead the finite model property fails, essentially because of the presence of an interpreted transitive relation (causality).

Still, hp-bisimilarity and the related logic has been shown to have good decidability properties. The equivalence itself is known to be decidable for finite safe Petri nets [38, 19, 24] (while hhp-bisimilarity is undecidable [21]). Additionally, the model-checking problem has been proved decidable for \mathcal{L}_{hp} over event structures satisfying a suitable regularity property [5].

Here we show that satisfiability for \mathcal{L}_{hp} is undecidable via a reduction from a well-known undecidable tiling problem [8], similarly to what was done for some two-variable logics [26]. Despite the first-order features of \mathcal{L}_{hp} , the reduction is not trivial since quantifications can be used in a quite restricted way: formulae can refer only to events enabled in the current configuration and execute them, inspecting their relations with (a limited number) of past events. In particular, it is impossible to relate events which are not consistent (in conflict). The “local” nature of quantifications makes it hard to constrain the event structure model to have a “grid shape”. Still, we can show that, given a domino system, it is possible to construct a formula which is satisfied only by event structures embedding an event-based representation of a valid tiling for such domino. Consequently, the formula is satisfiable if and only if the domino system admits a tiling, whence the undecidability of satisfiability.

We show that, instead, for \mathcal{L}_{hp}^f , the fragment of \mathcal{L}_{hp} without fixpoints, satisfiability is decidable. The result relies on the fact that \mathcal{L}_{hp}^f can be encoded into first-order logic and the fact that it enjoys the finite model property.

We foresee that our results can be of help for settling the decidability status of other similar logics for true concurrency, like other fragments of the logic in [3], or the event identifier logic of [30] and the mu-calculi for true concurrency in [18, 16, 17], for which satisfiability has not yet been investigated.



■ **Figure 1** A simple PES \mathcal{E} and (part of) its set of configurations.

2 Event Structures

We recap the basics of prime event structures [41], a widely known model of concurrency. Throughout the paper \mathbb{E} is a fixed countable set of events from which all events are taken, Λ a set of labels ranged over by a, b, c, \dots , and $\lambda : \mathbb{E} \rightarrow \Lambda$ a labelling function.

► **Definition 1** (prime event structure). A (Λ -labelled) prime event structure (PES) is a tuple $\mathcal{E} = \langle E, \leq, \# \rangle$, where $E \subseteq \mathbb{E}$ is the set of events and $\leq, \#$ are binary relations on E , called causality and conflict respectively, such that:

1. \leq is a partial order and $[e] = \{e' \in E \mid e' \leq e\}$ is finite for all $e \in E$;
2. $\#$ is irreflexive, symmetric and for all $e, e', e'' \in E$, if $e \# e' \leq e''$ then $e \# e''$.

Hereafter, we will assume that the components of a PES \mathcal{E} are named as in the definition above, possibly with subscripts. Concurrency is a derived relation, defined as follows.

► **Definition 2** (consistency, concurrency). Let \mathcal{E} be a PES. We say that $e, e' \in E$ are consistent, written $e \sim e'$, if $\neg(e \# e')$. A subset $X \subseteq E$ is called consistent if $e \sim e'$ for all $e, e' \in X$. We say that e and e' are concurrent, written $e \parallel e'$, if $e \sim e'$ and $\neg(e \leq e'), \neg(e' \leq e)$.

Causality and concurrency will be sometimes used on set of events. Given $X \subseteq E$ and $e \in E$, by $X < e$ we mean that for all $e' \in X, e' < e$. Similarly $X \parallel e$, resp. $X \sim e$, means that for all $e' \in X, e' \parallel e$, resp. $e' \sim e$.

A simple PES is depicted in Fig. 1(left). Graphically, curly lines represent immediate conflicts and the causal partial order proceeds along the arrows. Events are denoted by their labels, possibly with superscripts. For instance, in \mathcal{E} , the events a^1 and b^1 , labelled by a and b , respectively, are in conflict. Event a^0 causes the event b^0 which, in turn, is concurrent with each a^i and b^i (for $i \geq 1$).

A state of a system modelled as a PES is represented as the set of events executed to reach the state. It is formalised by the notion of configuration.

► **Definition 3** (configuration). Let \mathcal{E} be a PES. A configuration in \mathcal{E} is a finite consistent subset of events $C \subseteq E$ closed w.r.t. causality (i.e., $[e] \subseteq C$ for all $e \in C$). The set of finite configurations of \mathcal{E} is denoted by $\mathcal{C}(\mathcal{E})$.

In words, a configuration cannot contain events in conflict and it must be closed with respect to causality. The empty set of events \emptyset is always a configuration, which can be interpreted as the initial state of the computation. The evolution of a system can be represented by a transition system where configurations are states.

13:4 (Un)Decidability for History Preserving True Concurrent Logics

► **Definition 4** (transition system). *Let \mathcal{E} be a PES and let $C \in \mathcal{C}(\mathcal{E})$. Given $e \in E \setminus C$ such that $C \cup \{e\} \in \mathcal{C}(\mathcal{E})$, and $X, Y \subseteq C$ with $X < e$, $Y \parallel e$ we write $C \xrightarrow{X, \bar{Y} < e}_{\lambda(e)} C \cup \{e\}$, possibly omitting X , Y or the label $\lambda(e)$.*

Transitions are labelled by the executed event e . In addition, they can report its label $\lambda(e)$, a subset of causes X and a set of events $Y \subseteq C$ concurrent with e . When X or Y are empty they are normally omitted, e.g., we write $C \xrightarrow{X < e}_{\lambda(e)} C'$ for $C \xrightarrow{X, \emptyset < e}_{\lambda(e)} C'$ and $C \xrightarrow{e}_{\lambda(e)} C'$ for $C \xrightarrow{\emptyset, \emptyset < e}_{\lambda(e)} C'$. Some configurations of the PES \mathcal{E} in Fig. 1 (left) can be found in the same figure, on the right. Examples of transitions are $\{a^0, b^0\} \xrightarrow{a^0, \bar{b}^0 < b^1}_a \{a^0, b^0, b^1\}$ and $\{a^0, b^0\} \xrightarrow{a^0 < a^1}_a \{a^0, b^0, a^1\}$.

A PES is called image-finite when every configuration enables a finite number of events for each fixed label. In the rest of the paper all PESs will be assumed to be image-finite. This assumption, as it commonly happens for modal logics, is crucial to have a logical characterisation of bisimilarity in terms of a finitary logic.

► **Definition 5** (image-finiteness). *A PES \mathcal{E} is called image-finite when, for every configuration $C \in \mathcal{C}(E)$ and label $a \in \Lambda$, the set $\{C' \in \mathcal{C}(\mathcal{E}) \mid \exists e \in E. C \xrightarrow{e}_a C'\}$ is finite.*

3 A Logic for True Concurrency

We review the logic for concurrency of interest in the paper, a Hennessy-Milner style logic, originally introduced in [3], which corresponds to history-preserving bisimilarity. Its formulae predicate over executability of events in computations and their mutual relations (causality and concurrency).

Syntax

In order to specify dependencies between events in computation, formulae include event variables, from a fixed denumerable set Var , denoted by x, y, \dots . Tuples of variables like x_1, \dots, x_n will be denoted by the corresponding boldface letter \mathbf{x} and, abusing the notation, tuples will be often used as sets. The logic, besides standard propositional connectives, includes a diamond modality (and, dually, a box modality). The formula $\langle \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z \rangle \varphi$ holds when in the current configuration an \mathbf{a} -labelled event e is enabled which causally depends on the events bound to the variables in \mathbf{x} and is concurrent with those in \mathbf{y} . Event e is executed and bound to variable z , and then the formula φ must hold in the resulting configuration.

Fixpoint operators refer to propositional variables. In order to let them interact correctly with event variables, whose values can be passed from an iteration to the next one in the recursion, we use abstract propositions. For dealing with fixpoint operators we fix a denumerable set \mathcal{X}^a of *abstract propositions*, ranged over by X, Y, \dots . Each abstract proposition X has an arity $ar(X)$ and it represents a formula with $ar(X)$ (unnamed) free event variables. Then, for \mathbf{x} such that $|\mathbf{x}| = ar(X)$, we write $X(\mathbf{x})$ to indicate the abstract proposition X whose free event variables are named \mathbf{x} .

► **Definition 6** (hp-logic). *The syntax of \mathcal{L}_{hp} over the sets of event variables Var , abstract propositions \mathcal{X}^a and labels Λ is defined as follows:*

$$\begin{aligned} \varphi ::= & \top \mid \varphi \wedge \varphi \mid \langle \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z \rangle \varphi \mid (\mu Z(\mathbf{x}).\varphi)(\mathbf{y}) \mid Z(\mathbf{x}) \mid \\ & \text{F} \mid \varphi \vee \varphi \mid \llbracket \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z \rrbracket \varphi \mid (\nu Z(\mathbf{x}).\varphi)(\mathbf{y}) \end{aligned}$$

The free event variables of a formula φ are denoted $fv(\varphi)$ and defined in the obvious way. Just note that the modalities act as binders for the variable representing the event executed, hence $fv(\langle \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z \rangle \varphi) = fv(\llbracket \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z \rrbracket \varphi) = (fv(\varphi) \setminus \{z\}) \cup \mathbf{x} \cup \mathbf{y}$. The free propositions

in φ , i.e., the propositions not bound by μ , are denoted by $fp(\varphi)$. In fixpoint formulae, like $(\mu X(\mathbf{x}).\varphi)(\mathbf{y})$, we require that the tuple \mathbf{x} does not include multiple occurrences of the same variable and correspond exactly to the free event variables of the inner formula φ , i.e., $fv(\varphi) = \mathbf{x}$. Intuitively, the fixpoint part $\mu X(\mathbf{x}).\varphi$ defines a recursive formula $X(\mathbf{x})$ whose free variables are then instantiated with \mathbf{y} . The formula $(\mu X(\mathbf{x}).\varphi)(\mathbf{x})$ will be abbreviated as $\mu X(\mathbf{x}).\varphi$. When both $fv(\varphi)$ and $fp(\varphi)$ are empty we say that φ is *closed*. When \mathbf{x} or \mathbf{y} are empty they are often omitted, e.g., we write $\langle \mathbf{a} z \rangle \varphi$ for $\langle \emptyset, \bar{\emptyset} < \mathbf{a} z \rangle \varphi$.

Given a formula φ and variables $x, y \in Var$, we denote by $\varphi[y/x]$ the formula obtained from φ via a (capture avoiding) substitution of the free occurrences of x in φ by y . Similarly, given a proposition $Z(\mathbf{x}) \in \mathcal{X}$ and a formula ψ such that $fv(\psi) \subseteq \mathbf{x}$, we denote by $\varphi[Z(\mathbf{x}) := \psi]$ the formula obtained from φ by replacing free occurrences of $Z(\mathbf{y})$ by $\psi[\mathbf{y}/\mathbf{x}]$.

In the logic we can easily represent the possibility of performing concurrent events. Borrowing the notation from [3], we write $(\langle \mathbf{a} z \rangle \otimes \langle \mathbf{b} z' \rangle) \varphi$ for the formula $\langle \mathbf{a} z \rangle \langle \bar{z} < \mathbf{b} z' \rangle \varphi$ that declares the existence of two concurrent events labelled by \mathbf{a} and \mathbf{b} , respectively, such that if we execute such events and bind them to z and z' , respectively, then φ holds.

Consider again the PES in Fig. 1. Let $\psi_{2b} = (\langle \mathbf{b} x \rangle \otimes \langle \mathbf{b} y \rangle) \top$ be the formula stating that two concurrent \mathbf{b} -events can be executed. Then \mathcal{E} satisfies the formula $\langle \mathbf{a} z \rangle \psi_{2b}$, which states that after executing an \mathbf{a} -labelled event one can execute two concurrent \mathbf{b} -labelled events. It satisfies also the formula $\llbracket \mathbf{a} x \rrbracket (\nu X(x).(\psi_{2b} \wedge \llbracket x < \mathbf{a} z \rrbracket X(z)))$ stating that after any causal chain of \mathbf{a} -labelled events ψ_{2b} holds. Instead the formula $\mu X.(\llbracket \mathbf{a} x \rrbracket \otimes \llbracket \mathbf{a} y \rrbracket) \top \vee \langle \mathbf{a} z \rangle X$ that asks for the reachability of a state where two concurrent \mathbf{a} -labelled events can be executed, is false in \mathcal{E} . As a final example, the formula $\langle \mathbf{a} x \rangle (\nu X(x) \langle x < \mathbf{a} y \rangle X(y))$ asks for the existence of an infinite causal chain of \mathbf{a} -labelled events and it is satisfied by \mathcal{E} .

Semantics

Since the logic \mathcal{L}_{hp} is interpreted over PESs, the satisfaction of a formula is defined with respect to a configuration C , representing the state of the computation and an *environment* $\eta : Var \rightarrow E$, that binds free variables in the formula to events in C . Namely, if $Env_{\mathcal{E}}$ denotes the set of environments, the semantics of a formula will be a set of pairs in $\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$. Given a set of pairs $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ and two tuples of variables \mathbf{x} and \mathbf{y} , with $|\mathbf{x}| = |\mathbf{y}|$, we define $S[\mathbf{y}/\mathbf{x}] = \{(C, \eta') \mid \exists (C, \eta) \in S \wedge \eta(\mathbf{x}) = \eta'(\mathbf{y})\}$. The semantics of \mathcal{L}_{hp} also depends on a proposition environment providing a semantic interpretation for propositions.

► **Definition 7** (proposition environment). *Let \mathcal{E} be a PES. A proposition environment is a function $\pi : \mathcal{X} \rightarrow 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ such that for all abstract propositions X and tuples of variables \mathbf{x}, \mathbf{y} with $|\mathbf{x}| = |\mathbf{y}| = ar(X)$ it holds $\pi(X(\mathbf{y})) = \pi(X(\mathbf{x}))[\mathbf{y}/\mathbf{x}]$. The set of proposition environments, ranged by π , is denoted $PEnv_{\mathcal{E}}$.*

The condition posed on proposition environments ensures that the semantics of a formula only depends on the events that the environment associates with its free variables and that it does not depend on the naming of the variables.

We can now give the semantics of the logic \mathcal{L}_{hp} . Given an event environment η and an event e we write $\eta[x \mapsto e]$ to indicate the updated environment which maps x to e . Similarly, for a proposition environment π and $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, we write $\pi[Z(\mathbf{x}) \mapsto S]$ for the corresponding update. For a pair $(C, \eta) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ and variables $\mathbf{x}, \mathbf{y}, z$, we define the $(\mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z)$ -successors of (C, η) , as

$$\text{Succ}_{\mathcal{E}}^{\mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z}(C, \eta) = \{(C', \eta[z \mapsto e]) \mid C \xrightarrow{\eta(\mathbf{x}), \overline{\eta(\mathbf{y})} < e}_{\mathbf{a}} C'\}.$$

13:6 (Un)Decidability for History Preserving True Concurrent Logics

In words $\text{Succ}_{\mathcal{E}}^{\mathbf{x}, \bar{\mathbf{y}} < \mathbf{a}z}(C, \eta)$ consists of the pairs (C', η') where C' is a configuration reachable from C , by executing an event e satisfying the requirement expressed by $\mathbf{x}, \bar{\mathbf{y}} < \mathbf{a}z$, namely events in $\eta(\mathbf{x})$ are causes of e and events in $\eta(\mathbf{y})$ are concurrent with e . The environment η' is the update of η where event e has been bound to variable z .

► **Definition 8 (semantics).** *Let \mathcal{E} be a PES. The denotation of a formula in \mathcal{L}_{hp} is given by the function $\{\cdot\}_{\pi}^{\mathcal{E}} : \mathcal{L}_{hp} \rightarrow PEnv_{\mathcal{E}} \rightarrow 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ defined inductively as follows, where we write $\{\varphi\}_{\pi}^{\mathcal{E}}$ instead of $\{\varphi\}_{\pi}^{\mathcal{E}}(\pi)$:*

$$\begin{aligned} \{\top\}_{\pi}^{\mathcal{E}} &= \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} & \{\text{F}\}_{\pi}^{\mathcal{E}} &= \emptyset & \{Z(\mathbf{y})\}_{\pi}^{\mathcal{E}} &= \pi(Z(\mathbf{y})) \\ \{\varphi_1 \wedge \varphi_2\}_{\pi}^{\mathcal{E}} &= \{\varphi_1\}_{\pi}^{\mathcal{E}} \cap \{\varphi_2\}_{\pi}^{\mathcal{E}} & \{\varphi_1 \vee \varphi_2\}_{\pi}^{\mathcal{E}} &= \{\varphi_1\}_{\pi}^{\mathcal{E}} \cup \{\varphi_2\}_{\pi}^{\mathcal{E}} \\ \{\langle \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a}z \rangle \varphi\}_{\pi}^{\mathcal{E}} &= \{(C, \eta) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \mid \text{Succ}_{\mathcal{E}}^{\mathbf{x}, \bar{\mathbf{y}} < \mathbf{a}z}(C, \eta) \cap \{\varphi\}_{\pi}^{\mathcal{E}} \neq \emptyset\} \\ \{\llbracket \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a}z \rrbracket \varphi\}_{\pi}^{\mathcal{E}} &= \{(C, \eta) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \mid \text{Succ}_{\mathcal{E}}^{\mathbf{x}, \bar{\mathbf{y}} < \mathbf{a}z}(C, \eta) \subseteq \{\varphi\}_{\pi}^{\mathcal{E}}\} \\ \{\nu Z(\mathbf{x}).\varphi\}_{\pi}^{\mathcal{E}} &= \nu(f_{\varphi, Z(\mathbf{x}), \pi}) & \{\mu Z(\mathbf{x}).\varphi\}_{\pi}^{\mathcal{E}} &= \mu(f_{\varphi, Z(\mathbf{x}), \pi}) \end{aligned}$$

where $f_{\varphi, Z(\mathbf{x}), \pi} : 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}} \rightarrow 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ is the function defined by $f_{\varphi, Z(\mathbf{x}), \pi}(S) = \{\varphi\}_{\pi[Z(\mathbf{x}) \mapsto S]}^{\mathcal{E}}$, that we refer to as the semantic function of φ , $Z(\mathbf{x})$, π . Moreover, $\alpha(f_{\varphi, Z(\mathbf{x}), \pi})$, for $\alpha \in \{\mu, \nu\}$, denotes the corresponding (least or greatest) fixpoint. When $(C, \eta) \in \{\varphi\}_{\pi}^{\mathcal{E}}$ we say that the PES \mathcal{E} satisfies the formula φ in the configuration C and environments η, π .

The semantics of boolean connectives is standard. The formula $\langle \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a}z \rangle \varphi$ holds in (C, η) when configuration C enables an \mathbf{a} -labelled event e that is causally dependent on (at least) the events bound to the variables in \mathbf{x} and concurrent with (at least) those bound to the variables in \mathbf{y} and can be executed producing a new configuration $C' = C \cup \{e\}$ which, paired with the environment $\eta' = \eta[z \mapsto e]$, satisfies φ . The semantics of $\llbracket \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a}z \rrbracket \varphi$ is dual. When φ is closed (so that the environments η, π are irrelevant) and \mathcal{E} satisfies the formula φ in the empty configuration, we simply say that \mathcal{E} satisfies φ .

4 Undecidability of \mathcal{L}_{hp}

In this section we study the satisfiability problem for the logic \mathcal{L}_{hp} , i.e., the problem of determining whether a closed formula in \mathcal{L}_{hp} is satisfied by some (image-finite) PES. We prove it to be undecidable by reduction from domino tilings.

Domino systems

Tiling problems are a simple and general form of combinatorial problems introduced in [39, 40] for proving the unsolvability of the $\forall\exists\forall$ -prefix class in the pure predicate calculus. Along the years they revealed to be a powerful tool for proving undecidability results for fragments of first-order logic and for decision problems in mathematical theories (see, e.g., [10]).

► **Definition 9 (dominoes).** *A domino system is a tuple $\mathcal{D} = (D, H, V)$ where D is a finite set and $H, V \subseteq D^2$ are binary relations.*

The elements of D should be thought of as square tiles (called domino pieces, or *Wang tiles*) with a color on each side and a fixed orientation. Hence for $d, e \in D$ read $(d, e) \in V$, or dVe , as “ e can stand immediately above d ” because the upper color of d and the lower color of e are the same. Similarly $(d, e) \in H$, or dHe , should be read as “the left side of e can be attached to the right side of d ”. A tiling is thus a covering of $\mathbb{N} \times \mathbb{N}$ thought as an infinite board on which each point is a spot to be occupied by a domino.

The plane $\mathbb{N} \times \mathbb{N}$ together with the binary relations $H = \{(p, q), (p+1, q) : p, q \in \mathbb{N}\}$ and $V = \{(p, q), (p, q+1) : p, q \in \mathbb{N}\}$ can be seen as domino system referred to as the *grid* $\mathfrak{S}_{\mathbb{N}}$. Given two domino systems $\mathcal{D} = (D, H, V)$ and $\mathcal{D}' = (D', H', V')$ a homomorphism $f : \mathcal{D} \rightarrow \mathcal{D}'$ is a function $f : D \rightarrow D'$ such that for all $(d_1, d_2) \in H$ it holds that $(f(d_1), f(d_2)) \in H'$ and, similarly, for all $(d_1, d_2) \in V$ it holds that $(f(d_1), f(d_2)) \in V'$. Tilings can be then formalised relying on the notion of homomorphism.

► **Definition 10** (tiling). *A tiling of a domino system \mathcal{D} is a homomorphism $T : \mathfrak{S}_{\mathbb{N}} \rightarrow \mathcal{D}$.*

Given a tiling $T : \mathfrak{S}_{\mathbb{N}} \rightarrow \mathcal{D}$, intuitively $T(p, q) = d$ means that the point (p, q) is filled with a copy of the piece d . Our undecidability proof relies on the following well-known result about domino tilings.

► **Theorem 11** (undecidability of tiling [8]). *The problem of establishing the existence of a tiling for a given domino system is undecidable.*

Reduction from Domino Tiling

In order to show that \mathcal{L}_{hp} is undecidable, we associate with each domino system a formula of \mathcal{L}_{hp} such that the formula is satisfiable if and only if the domino system admits a tiling.

For various fragments of first-order logic (even with only two variables), the approach consists in using suitable unary predicates to establish a correspondence between elements of the model and domino pieces, and binary predicates to represent adjacency. For example, undecidability results for two-variables logics are obtained in [26, 22] using in a crucial way the transitivity of some predicates in the chosen relational vocabulary. A suitable interplay between existential and universal quantifications allows to build formulae whose models are *grid-like*, i.e., such that $\mathfrak{S}_{\mathbb{N}}$ can be embedded into these models.

Similarly, the idea here is to embed the grid inside a PES. Events are associated, via their label, with pieces of the domino and adjacency is suitably represented with arrangements of causality and concurrency between events. This is far from trivial since formulae of the logic can refer only to events enabled in the current configuration and execute them, checking their relations with (a limited number) of past events. As a consequence, quantification has a “local” nature and only the relations between consistent events can be inspected. This is quite a subtle point, since similar restrictions upon quantifications can yield decidable logics, like the guarded fragment with transitive guards studied in [35].

Greatest fixpoints like $(\nu X(\mathbf{x}).\varphi)(\mathbf{y})$ can be used to ensure that what is being predicated by $\varphi(\mathbf{x})$ holds repeatedly throughout an unbounded number of reachable configurations. For a given domino system, we define formulae that verify the proper adjacency of pieces by exploring the events along diagonal slices of the grid, starting from the bottom row and ending at the leftmost column. All slices are finite causal chains but overall they grow unboundedly in length. The formula for a domino system is quite complex and not immediate to read, but the underlying intuition will be explained in detail after the definition.

► **Definition 12** (formula for a domino). *Let \mathcal{D} be a domino system with $D = \{d_1, \dots, d_n\}$. Define the set of $6n$ labels $A = \{b_k^s \mid b \in \{a, i, j\} \wedge k \in \{1, \dots, n\} \wedge s \in \{0, 1\}\}$. For $s \in \{0, 1\}$ we let \bar{s} abbreviate $1 - s$. Consider the following formulae (1)–(4).*

$$\bigvee_{\substack{d_k \text{ H } d_h \\ d_k \text{ V } d_v}} \langle i_k^0 x \rangle \langle x < i_h^1 y \rangle \langle y < j_v^1 z \rangle \top \quad (1)$$

$$\begin{aligned}
 & \bigwedge_{\substack{b \in \{a, j\} \\ d_k, d_l \in D \\ s \in \{0, 1\}}} \llbracket i_k^s x \rrbracket \llbracket x < b_l^s y \rrbracket (\nu X(x, y, z)). \\
 & \bigvee_{d_k H d_h} \langle x, \bar{y} < i_h^{\bar{s}} u \rangle \top \wedge \bigwedge_{\substack{c \in \{a, j\} \\ d_m \in D}} \llbracket z < c_m^s v \rrbracket X(x, y, v))(x, y, y)
 \end{aligned} \tag{2}$$

$$\begin{aligned}
 & \bigwedge_{\substack{b \in \{a, i\} \\ c \in \{a, j\} \\ d_l, d_m \in D \\ d_k H d_h \\ s \in \{0, 1\}}} \llbracket b_k^s x \rrbracket \llbracket x < a_l^s y \rrbracket \llbracket y < c_m^s z \rrbracket \llbracket x, \bar{y} < b_h^{\bar{s}} w \rrbracket (\nu X(y, z, w, u)). \\
 & \bigvee_{\substack{d_k V d_p \\ d_l H d_p}} \langle y, w, \bar{z} < a_p^{\bar{s}} r \rangle \top \wedge \bigwedge_{\substack{e \in \{a, j\} \\ d_q \in D}} \llbracket u, \bar{w} < e_q^s v \rrbracket X(y, z, w, v))(y, z, w, z)
 \end{aligned} \tag{3}$$

$$\begin{aligned}
 & \bigwedge_{\substack{b \in \{a, i\} \\ d_l \in D \\ d_k H d_h \\ s \in \{0, 1\}}} \llbracket b_k^s x \rrbracket \llbracket x < j_l^s y \rrbracket \llbracket x, \bar{y} < b_h^{\bar{s}} z \rrbracket \bigvee_{\substack{d_k V d_p \\ d_l H d_p \\ d_l V d_q}} \langle y, z < a_p^{\bar{s}} v \rangle \langle v < j_q^{\bar{s}} w \rangle \top
 \end{aligned} \tag{4}$$

Calling ψ_i the corresponding formula (i) above, we denote the formula for the domino system \mathcal{D} by $\varphi_{\mathcal{D}} = \psi_1 \wedge \nu Z.(\psi_2 \wedge \psi_3 \wedge \psi_4 \wedge \llbracket A z \rrbracket Z)$, where we write $\llbracket A z \rrbracket Z$ for $\bigwedge_{b_k^s \in A} \llbracket b_k^s z \rrbracket Z$.

Intuitively, the formula $\varphi_{\mathcal{D}}$ requires a model to contain a grid of consistent events, as depicted in Fig. 2. The formula also arranges causal dependencies that constrain the order of exploration, i.e., of execution, of the grid as an ever growing right-angled triangle. This allows to build the grid one diagonal at a time, starting from the leftmost smallest diagonal which consists of a single event at coordinates (0, 0) in Fig. 2. Every diagonal, except the first, is delimited by two events with special labels: the first at the bottom of the diagonal is labelled i , the last at the top is labelled j . Inner events are, instead, all labelled a . Actually, labels include also a subscript and a superscript. Subscripts $k \in \{1, \dots, n\}$ represent the associated domino piece d_k . Superscripts $s \in \{0, 1\}$, instead, are used to distinguish events in a diagonal from those in the next and previous ones. So the superscript for all the events in a diagonal starting at coordinates $(t, 0)$ is simply $s = t \bmod 2$, as shown in the figure.

To explain how the formula works we first comment on how the satisfaction of the formula implies the existence of a grid in the model. For this, the superscripts on labels play no role and can be safely ignored. They will become relevant later for the converse implication.

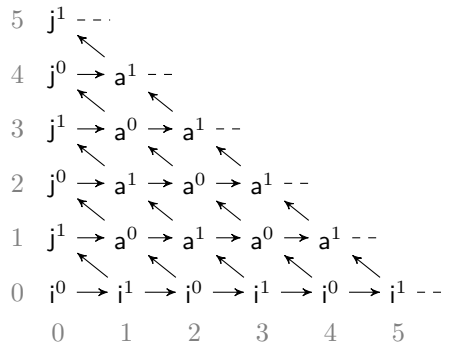
The first two diagonals are determined by the first subformula ψ_1 (1). This formula simply requires the existence of three events, executable from the initial state, such that each one causes the next. Such events represent the first three domino pieces in the tiling. So, they are required to be labelled i_k, i_h, j_v , in a way that the adjacency constraints are respected, i.e., $d_k H d_h$ and $d_k V d_v$. Each diagonal beyond the first two is, instead, jointly defined by the other three subformulae ψ_2, ψ_3, ψ_4 , which are guaranteed to be checked on every reachable state of the computation via the outermost greatest fixpoint of $\varphi_{\mathcal{D}}$.

In Fig. 3 we give a graphical representation of such three subformulae. In each sub-figure, greyed out events represent the configuration to which the subformula is intended to apply. Events highlighted in red are universally quantified by box operators in the formula, while events highlighted in blue are existentially quantified by diamond operators. The picture also reports, close to events, the variables to which such events are bound in the formula.

The subformula ψ_2 (2) starts a new diagonal. The first event in the new diagonal must be labelled i_p , for some domino piece d_p horizontally compatible with the one corresponding to the first event in the previous diagonal, labelled i_l in the figure. Moreover, the new event i_p must be caused by i_l and concurrent with the second event in the previous diagonal, labelled a_m in the figure. In addition, using a fixpoint subformula, we ask that the new event is executed after (hence consistent with) the whole red diagonal. This implies that i_p is necessarily concurrent with all the events in the red diagonal, although not explicitly required. In fact, since it is consistent and executed after the diagonal, the only alternative would be that i_p were caused by some event in the diagonal. But this would mean that also a_m causes i_p , while we know that they are concurrent.

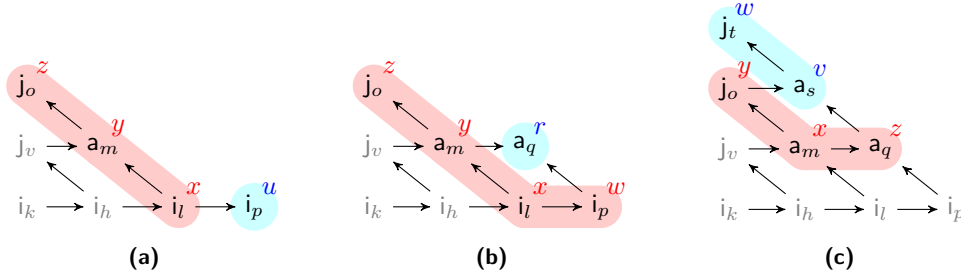
The subformula ψ_3 (3) continues the “construction” of the new diagonal starting from the second event and, with each successive application, up to the third-to-last event of the new diagonal. Basically, it builds the a -labelled part of the new diagonal, except the last a -labelled event. In particular, a single application of ψ_3 ensures the executability of an event labelled a_q , for some domino piece d_q vertically compatible with the one corresponding to the event just below in the grid, which is necessarily part of the previous diagonal, labelled i_l in the figure, and horizontally compatible with the one just after the latter in the previous diagonal, that is a_m in the example. The new event a_q must be caused by the one just before it in the new diagonal, i_p in this case. Furthermore, as in the previous property, the new event must be caused by the event a_m at the same height in the previous diagonal, and concurrent with the events coming after it in such diagonal.

Finally, the subformula ψ_4 (4) concludes the construction of the new diagonal, requiring the executability of the last two events labelled a_s and j_t , respectively. As before, the corresponding domino pieces d_s and d_t must be compatible with those of the adjacent events below and on the left, all of which belonging to the previous diagonal. Moreover, the new event a_s must be caused by the one just before it in the new diagonal and by the last event of the previous diagonal. Instead, j_t is just required to be caused by a_s , which, however, by transitivity of the causality relation, means that j_t is caused by every other event in the new diagonal and all those in the previous diagonals.



■ **Figure 2** Grid of events for the domino tiling (with simplified event labels).

13:10 (Un)Decidability for History Preserving True Concurrent Logics



■ **Figure 3** Graphical representation of the properties, from left to right, (2), (3) and (4).

Note that the subformulae ψ_2 and ψ_3 use a (greatest) fixpoint in order to fully explore a diagonal, which is unbounded. This ensures the consistency of each newly added event with the previous diagonal and thus with the whole right-angled triangle up to such diagonal.

Relying on the intuitions described above we can prove the desired result: given a domino system \mathcal{D} , the formula $\varphi_{\mathcal{D}}$ is satisfiable if and only if \mathcal{D} admits a tiling. We next present a sketch of the proof.

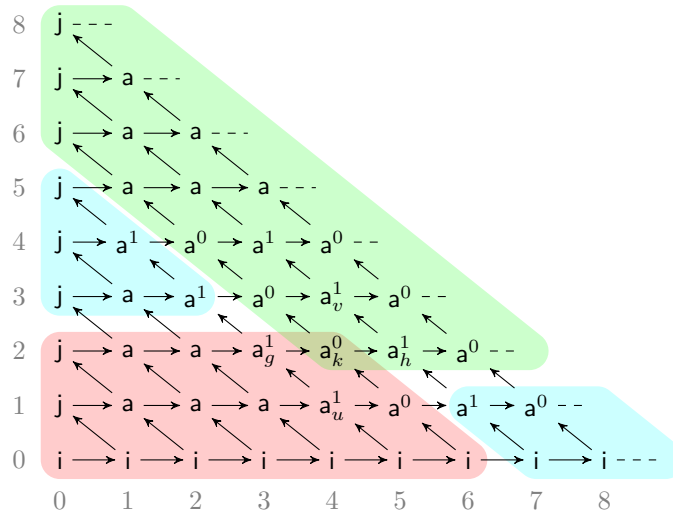
For proving the first implication we show how, given a PES which satisfies the formula $\varphi_{\mathcal{D}}$, one can build a tiling for the domino system \mathcal{D} . Recall that a tiling of \mathcal{D} is a function $T : \mathbb{N} \times \mathbb{N} \rightarrow D$ complying with the adjacency relations H and V of \mathcal{D} . Proceeding, as mentioned before, by diagonals, we can inductively define an infinite chain of functions f_i , for all $i \in \mathbb{N}^+$, whose domain is the right-angled triangle up to the $(i + 1)$ -th diagonal, i.e. $\{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x + y \leq i\}$. The first function f_1 is obtained directly from the events guaranteed to exist by the satisfaction of the subformula ψ_1 of $\varphi_{\mathcal{D}}$. Every other function f_i is defined extending f_{i-1} and using the events whose existence is required by the other subformulae of $\varphi_{\mathcal{D}}$. The join of the f_i 's is defined on the whole grid and provides a tiling.

► **Theorem 13** (satisfiability implies tiling). *Let \mathcal{D} be a domino system with $D = \{d_1, \dots, d_n\}$, if the corresponding formula $\varphi_{\mathcal{D}}$ is satisfiable, then \mathcal{D} admits a tiling.*

For the converse implication, we need to show how to transform a tiling T of \mathcal{D} into a PES which satisfies the property $\varphi_{\mathcal{D}}$. From the grid of domino pieces corresponding to the tiling T we define a PES \mathcal{E} whose events are $E = \mathbb{N} \times \mathbb{N}$. Events are labelled i_k^s if they belong to the bottom row, j_k^s if in the left-most column, except $(0, 0)$, or a_k^s otherwise, where k is the subscript of the corresponding domino piece d_k occupying the same position, and s is the index corresponding to the diagonal to which the event belongs. Explicitly, if the coordinates of the event are (x, y) , then the index is $s = (x + y) \bmod 2$, hence it is the same for all the events in a same diagonal. The PES has empty conflict relation, while causality is defined as in Fig. 4. In this way, every event

- is caused exactly by those at lower or equal height in the smallest right-angled triangle containing the event,
- causes those at higher or equal height outside of such triangle or above the event along its diagonal,
- is concurrent with all the others.

For instance, consider the event at position $(4, 2)$ in Fig. 4, labelled a_k^0 . It is caused by the events highlighted in red, it causes those highlighted in green, and it is concurrent with those highlighted in blue. It is easy to see that \mathcal{E} is well-defined, i.e. it is a PES, in fact, the set of causes of each event is clearly finite. Moreover, \mathcal{E} is image-finite since every configuration enables a finite number of events (bounded by 1 plus half the size of the configuration).



■ **Figure 4** PES for the canonical grid of the domino tiling (with simplified event labels).

Formalising the ideas outlined above, one can prove that $\varphi_{\mathcal{D}}$ holds in the initial state of \mathcal{E} . In particular, the formulae ψ_2 , ψ_3 and ψ_4 , inside the outermost fixpoint, can be shown to hold in every reachable state, sometimes vacuously. For example, consider the configuration C consisting of all the events appearing in Fig. 4 except those highlighted in green. Let us focus on the subformula ψ_3 (3) and argue that it holds in configuration C . Observe that the initial three box modalities in ψ_3 require a specific structure to be executable, in absence of which the formula holds vacuously. Such structure consists of a causal chain of three events, labelled with some combination of letters i, a, j but all with the same superscript s . Inspecting the structure in the figure, it occurs that there are only two possible causal chains of three events executable from C : both starting with $(4, 2)$, and then going either along its diagonal up to $(2, 4)$, or along its row up to $(6, 2)$. However, since the three events must be labelled with the same superscript s , only the chain along the diagonal is actually considered by the formula (along rows events alternate superscripts instead). Then, exploiting the similarities with the graphical representation of ψ_3 in Fig. 3b, it is possible to see that after binding those events the rest of the property holds.

► **Theorem 14** (tiling implies satisfiability). *Let \mathcal{D} be a domino system with $D = \{d_1, \dots, d_n\}$, if \mathcal{D} admits a tiling, then the formula $\varphi_{\mathcal{D}}$ is satisfiable.*

By the theorems above and the undecidability of the domino problem we conclude.

► **Corollary 15** (undecidability of \mathcal{L}_{hp}). *The satisfiability problem for \mathcal{L}_{hp} is undecidable.*

5 Decidability without fixpoints

In this section we show that, in absence of fixpoint operators, the logic \mathcal{L}_{hp} has the finite model property. Moreover, we provide an encoding of formulae into first-order logic preserving their (un)satisfiability. As a consequence we deduce tha satisfiability for the logic without fixpoints becomes decidable.

In the following we will denote by \mathcal{L}_{hp}^f the fragment of the logic \mathcal{L}_{hp} without fixpoint operators (and propositions). Consequently, the semantics of formulae in \mathcal{L}_{hp}^f can be defined without proposition environments π , since there are no propositions to interpret.

13:12 (Un)Decidability for History Preserving True Concurrent Logics

In order to prove the finite model property the idea consists in showing that every model of a formula can be reduced to a finite one by restricting to a suitable chosen subset of events. To this aim, we introduce a way to truncate PESs by keeping only events up to a certain causal level and with a specific subset of labels. The causal level of an event e is inductively defined as $lev(e) = \max\{lev(e') + 1 \mid e' \in E \wedge e' < e\}$, where it is intended that $\max \emptyset = 0$.

► **Definition 16** (prefix of a PES). *Let E be a PES. For $k \in \mathbb{N}$ and $A \subseteq \Lambda$, consider the set of events $E^{(A,k)} = \{e \in E \mid lev(e) \leq k \wedge \forall e' \in [e]. \lambda(e') \in A\}$. Then, the A -labelled k -prefix of \mathcal{E} is the PES defined as $\mathcal{E}^{(A,k)} = \langle E^{(A,k)}, <|_{E^{(A,k)}}, \#|_{E^{(A,k)}} \rangle$.*

Note that, by the very definition of causal level, $lev(e') < lev(e)$ for all $e' < e$; hence, the (A -labelled) k -prefix $\mathcal{E}^{(A,k)}$ of a PES \mathcal{E} is a causally closed subset of \mathcal{E} . From this observation, it immediately follows that $\mathcal{E}^{(A,k)}$ is indeed a PES, i.e., the definition is well-given.

Notably, when a PES is image-finite, the same holds for all its prefixes. Hence, for every $k \in \mathbb{N}$ and finite A , the A -labelled k -prefix $\mathcal{E}^{(A,k)}$ can be shown to be finite.

► **Lemma 17** (finiteness of prefixes). *Let \mathcal{E} be a image-finite PES. For all $k \in \mathbb{N}$ and $A \subseteq \Lambda$, if A is finite, then $\mathcal{E}^{(A,k)}$ is finite.*

Now, in order to prove the finite model property of \mathcal{L}_{hp}^f it is enough to show that the satisfaction of formulae of \mathcal{L}_{hp}^f is preserved when truncating a PES up to a suitable level k and set of labels A . Both k and A can be obtained directly from the formula. Let the *modal depth* of a formula φ , denoted by $d(\varphi)$, be defined as usual. If φ is \top or F , its modal depth is 0. If it is a conjunction or disjunction, the modal depth is the maximum of those of the conjuncts, resp. disjuncts. Otherwise, when φ consists of a modality followed by a subformula ψ , the modal depth is $d(\varphi) = 1 + d(\psi)$. Let $A(\varphi)$ be the (finite) set of labels appearing in the formula φ . Then, whenever a formula φ is satisfied by a PES \mathcal{E} , it is also satisfied by the $A(\varphi)$ -labelled $d(\varphi)$ -prefix of \mathcal{E} , which, by the previous lemma, is finite.

► **Theorem 18** (finite model property of \mathcal{L}_{hp}^f). *Let φ be a closed formula of \mathcal{L}_{hp}^f . If φ is satisfiable, then there exists a finite PES satisfying φ .*

The result above implies that satisfiability for \mathcal{L}_{hp}^f is semi-decidable. In fact, finite PESs are denumerable and checking whether a finite PES satisfies a formula is decidable. Then, to conclude it is sufficient to observe that the axioms of PESs are expressible as first-order formulae and \mathcal{L}_{hp}^f , as it happens for many modal logics, can be encoded into first-order logic, hence also unsatisfiability is semi-decidable.

First, as mentioned, for a fixed formula in \mathcal{L}_{hp}^f the set of labels appearing in it is finite. Once the finite set of labels A is fixed, the theory of prime event structures, apart from the axiom of finite causes, is expressible as a finite set of first-order axioms.

► **Definition 19** (first-order theory of PESs). *Let $A \subseteq \Lambda$ be a finite set of labels. The first order theory of theory of PESs over A , consists of the following axioms with $<$ and $\#$ as binary predicates and labels as unary predicates.*

1. $\forall x, y, z. (x < y) \wedge (y < z) \rightarrow (x < z)$
2. $\forall x. \neg(x < x)$
3. $\forall x. \neg(x \# x)$
4. $\forall x, y. (x \# y) \rightarrow (y \# x)$
5. $\forall x, y, z. (x < y) \wedge (x \# z) \rightarrow (y \# z)$
6. $\forall x. \left(\bigvee_{a \in A} a(x) \wedge \bigwedge_{a, b \in A, a \neq b} \neg(a(x) \wedge b(x)) \right)$

We denote by $T_{PES}(A)$ the conjunction of the above axioms.

Axioms (1) and (2) state that $<$ is a (strict) partial order. Axioms (3) and (4) ask conflict $\#$ to be irreflexive and symmetric, while (5) requires conflict to be inherited along causality. Finally (6) asks that each event has exactly one label.

Now, given a set of variables $X \subseteq \text{Var}$ and a variable $y \in \text{Var}$, let us write $y \in X$ as an abbreviation for $\bigvee_{x \in X} (x = y)$. Then we can express the property of being a configuration as:

$$\begin{aligned} \text{conf}(X) \equiv & \forall x, y. ((x \in X) \wedge (y < x) \rightarrow (y \in X)) \wedge \\ & \forall x, y. ((x \# y) \rightarrow \neg((x \in X) \wedge (y \in X))) \end{aligned}$$

We can finally provide the translation of \mathcal{L}_{hp}^f into first-order formulae.

► **Definition 20** (compiling \mathcal{L}_{hp}^f to first-order logic). *Let X be a finite set of variables and let φ be a formula of \mathcal{L}_{hp}^f . We denote by $(\varphi)_X$ the first-order formula inductively defined as follows:*

- $(\top)_X = \top$ and $(\text{F})_X = \text{F}$
- $(\varphi \wedge \psi)_X = (\varphi)_X \wedge (\psi)_X$ and $(\varphi \vee \psi)_X = (\varphi)_X \vee (\psi)_X$
- $(\langle \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z \rangle \varphi)_X = \exists z. \bigwedge_{x \in \mathbf{x}} (x < z) \wedge \bigwedge_{y \in \bar{\mathbf{y}}} \neg(y < z) \wedge \mathbf{a}(z) \wedge \neg(z \in X) \wedge \text{conf}(X \cup \{z\}) \wedge (\varphi)_{X \cup \{z\}}$
- $(\llbracket \mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z \rrbracket \varphi)_X = \forall z. (\bigwedge_{x \in \mathbf{x}} (x < z) \wedge \bigwedge_{y \in \bar{\mathbf{y}}} \neg(y < z) \wedge \mathbf{a}(z) \wedge \neg(z \in X) \wedge \text{conf}(X \cup \{z\})) \rightarrow (\varphi)_{X \cup \{z\}}$

Then, given a closed formula φ of the logic \mathcal{L}_{hp}^f we can obtain an equisatisfiable first-order formula by taking the conjunction of the first order theory of PESs and the encoding of φ defined above.

► **Proposition 21.** *Let φ be a closed formula of \mathcal{L}_{hp}^f and let A be the set of labels occurring in φ . It holds that φ is satisfiable iff the first-order formula $T_{PES}(A) \wedge (\varphi)_\emptyset$ is satisfiable.*

The proof is straightforwardly based on the observation that a PES \mathcal{E} satisfying φ can be seen as a first-order structure satisfying $T_{PES}(A) \wedge (\varphi)_\emptyset$, and vice versa. The only delicate aspects is the absence of the axiom of finite causes in $T_{PES}(A)$. Hence, it could happen that $T_{PES}(A) \wedge (\varphi)_\emptyset$ is satisfiable by a structure which, seen as a PES \mathcal{E} , includes events with infinitely many causes. However, in this case, since these events would never be executable, it is clear that also the PES \mathcal{E}' obtained from \mathcal{E} by removing all events with infinitely many causes is a model for φ .

We can finally deduce that the satisfiability for \mathcal{L}_{hp}^f is decidable.

► **Corollary 22** (decidability of \mathcal{L}_{hp}^f). *The satisfiability problem for the logic fragment \mathcal{L}_{hp}^f is decidable and every satisfiable formula has a finite model.*

The proof combines the results proved above. First, by Theorem 18, when a formula φ in \mathcal{L}_{hp}^f is satisfiable it has a finite model labelled over the finite set A of labels occurring in φ . Since the finite PESs labelled over a finite alphabet are denumerable and checking whether a finite PES satisfies a formula is decidable, we can semi-decide satisfiability of a formula φ by enumerating the finite PESs labelled over A and checking whether each of the generated PES satisfies φ . Moreover, by Proposition 21, unsatisfiability of a formula φ in \mathcal{L}_{hp}^f is reducible to unsatisfiability of the first-order formula $T_{PES}(A) \wedge (\varphi)_\emptyset$, and thus it is semi-decidable. We conclude that satisfiability for \mathcal{L}_{hp}^f is decidable.

6 Conclusions and Perspectives

The logic \mathcal{L}_{hp} investigated in this paper is one of a number of fragments of a logic introduced in [3]. Other fragments \mathcal{L}_p and \mathcal{L}_s can be obtained by syntactical restrictions, characterising coarser true concurrent notions of bisimilarity, namely pomset and step bisimilarity. The full logic, instead, corresponds to hereditary hp-bisimilarity, the finest behavioural equivalence in the true concurrent spectrum of [37], finer than hp-bisimilarity. Each logic fragment admits a variant with fixpoints and a variant without fixpoints. However, the induced logical equivalences for image-finite PESs are the same with or without fixpoints. We proved that the satisfiability problem for \mathcal{L}_{hp} is undecidable, and thus the same holds also for the logic for hereditary hp-bisimilarity in [3]. On the other hand, satisfiability is decidable for \mathcal{L}_{hp}^f , the fragment of \mathcal{L}_{hp} without fixpoints.

Some preliminary investigations suggest that the step logic \mathcal{L}_s (with fixpoints) is decidable via a reduction to the propositional μ -calculus. The same technique appears to be promising for the pomset logic \mathcal{L}_p (with fixpoints) but this case is more complex and unresolved to this day. Similar logics for true concurrent properties are event identifier logic of [30] and the mu-calculi for true concurrency in [18, 16, 17]. Also in this case, to the best of our knowledge satisfiability has not yet been investigated. This offers a range of open questions that, when answered, would draw an interesting picture of problems across the decidability border.

In a sense there are “two dimensions” to the satisfiability problem: one is the syntax and the other the semantics, so that there are also many interesting variants and facets of the satisfiability question when the restrictions are imposed on the model side. For example a notable semantics is that of *regular* models in the sense of [36]. Investigating whether restricting the semantics with the constraint of regularity affects decidability is an intriguing direction of future work.

A formalisation of the semantics of the logics in terms of suitable (parity) games is often a source of inspiration for facing complexity and decidability issues for modal logics. Currently, there is no established game-theoretical characterisation of the semantics of the logic in [3], of which \mathcal{L}_{hp} is a fragment. However, such a development could be naturally guided by the approach in [16, 17] and by the relation between fixpoint games [4] and logics for concurrency, hinted at in [28]. This also appears as an interesting route to explore.


References

- 1 Rajeev Alur, Doron A. Peled, and Wojciech Penczek. Model-checking of causality properties. In *Proceedings of LICS'95*, pages 90–100. IEEE Computer Society, 1995.
- 2 Paolo Baldan and Alberto Carraro. A causal view on non-interference. *Fundamenta Informaticae*, 140(1):1–38, 2015.
- 3 Paolo Baldan and Silvia Crafa. A logic for true concurrency. *Journal of the ACM*, 61(4):24:1–24:36, 2014.
- 4 Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. Fixpoint games in continuous lattices. *PACMPL*, 3(POPL):26:1–26:29, 2019.
- 5 Paolo Baldan and Tommaso Padoan. Local model checking in a logic for true concurrency. In Javier Esparza and Andrzej S. Murawski, editors, *Proceedings of FoSSaCS'17*, volume 10203 of *LNCS*, pages 407–423. Springer, 2017.
- 6 Paolo Baldan and Tommaso Padoan. Model checking a logic for true concurrency. *ACM Trans. Comput. Log.*, 21(4):34:1–34:49, 2020.
- 7 Marek A. Bednarczyk. Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical report, Polish Academy of Sciences, 1991.


- 8 Robert Berger. *The Undecidability of the Domino Problem*. Memoirs ; No 1/66. American Mathematical Society, 1966.
- 9 Eike Best, Raymond Devillers, Astrid Kiehn, and Lucia Pomello. Fully concurrent bisimulation. *Acta Informatica*, 28:231–261, 1991.
- 10 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Universitext. Springer Berlin Heidelberg, 2001.
- 11 J. Bradford and S. Fröschle. Independence-friendly modal logic and true concurrency. *Nordic Journal of Computing*, 9(1):102–117, 2002.
- 12 R. De Nicola and G. Ferrari. Observational logics and concurrency models. In K. V. Nori and C. E. V. Madhavan, editors, *Proceedings of FST-TCS'90*, volume 472 of *LNCS*, pages 301–315. Springer, 1990.
- 13 Pierpaolo Degano, Rocco De Nicola, and Ugo Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In Jaco W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop*, volume 354 of *LNCS*, pages 438–466, Heidelberg, DE, 1988. Springer.
- 14 Marlon Dumas and Luciano García-Bañuelos. Process mining reloaded: Event structures as a unified representation of process models and event logs. In R. R. Devillers and A. Valmari, editors, *Petri Nets 2015*, volume 9115 of *LNCS*, pages 33–48. Springer, 2015.
- 15 Azadeh Farzan and P. Madhusudan. Causal atomicity. In T. Ball and R. B. Jones, editors, *Proceedings of CAV'06*, volume 4144 of *LNCS*, pages 315–328, 2006.
- 16 Julian Gutierrez. Logics and bisimulation games for concurrency, causality and conflict. In L. de Alfaro, editor, *Proceedings of FoSSaCS'09*, volume 5504 of *LNCS*, pages 48–62. Springer, 2009.
- 17 Julian Gutierrez. *On bisimulation and model-checking for concurrent systems with partial order semantics*. PhD thesis, University of Edinburgh, 2011.
- 18 Julian Gutierrez and Juilian C. Bradford. Model-checking games for fixpoint logics with partial order models. In M. Bravetti and G. Zavattaro, editors, *Proceedings of CONCUR'09*, volume 5710 of *LNCS*, pages 354–368. Springer, 2009.
- 19 Lalita Jategaonkar and Albert R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, 1996.
- 20 Alan Jeffrey and James Riely. On thin air reads towards an event structures model of relaxed memory. In M. Grohe, E. Koskinen, and N. Shankar, editors, *Proceedings of LICS'16*, pages 759–767. ACM, 2016.
- 21 Marcin Jurdzinski, Mogens Nielsen, and Jiri Srba. Undecidability of domino games and hhp-bisimilarity. *Information and Computation*, 184(2):343–368, 2003.
- 22 Emanuel Kieronski. Results on the guarded fragment with equivalence or transitive relations. In C.-H. Luke Ong, editor, *Computer Science Logic*, volume 3634 of *LNCS*, pages 309–324. Springer, 2005.
- 23 P. Madhusudan. Model-checking trace event structures. In *Proceedings of LICS 2013*, pages 371–380. IEEE Computer Society, 2003.
- 24 Ugo Montanari and M. Pistore. Minimal transition systems for history-preserving bisimulation. In R. Reischuk and M. Morvan, editors, *Proceedings of STACS'97*, volume 1200 of *LNCS*, pages 413–425. Springer, 1997.
- 25 Mogens Nielsen and Christian Clausen. Games and logics for a noninterleaving bisimulation. *Nordic Journal of Computing*, 2(2):221–249, 1995.
- 26 Martin Otto. Two variable first-order logic over ordered domains. *Journal of Symbolic Logic*, 66:685–702, 1999.
- 27 Tommaso Padoan. Relating some logics for true concurrency. In Alessandro Aldini and Marco Bernardo, editors, *Proceedings of ICTCS '18*, volume 2243 of *CEUR Workshop Proceedings*, pages 242–253. CEUR-WS.org, 2018.
- 28 Tommaso Padoan. *Tableaux, Automata and Games for True Concurrency Properties*. PhD thesis, University of Padua, 2019.

- 29 Wojciech Penczek. Branching time and partial order in temporal logics. In *Time and Logic: A Computational Approach*, pages 179–228. UCL Press, 1995.
- 30 I. Phillips and I. Ulidowski. Event identifier logic. *Mathematical Structures in Computer Science*, 24(2):1–51, 2014. doi:10.1017/S0960129513000510.
- 31 Jean Pichon-Pharabod and Peter Sewell. A concurrency semantics for relaxed atomics that permits optimisation and avoids thin-air executions. In R. Bodík and R. Majumdar, editors, *Proceedings of POPL'16*, pages 622–633. ACM, 2016.
- 32 S. Pinchinat, F. Laroussinie, and Ph. Schnoebelen. Logical characterization of truly concurrent bisimulation. Technical Report 114, LIFIA-IMAG, Grenoble, 1994.
- 33 Christian Prisacariu. Higher dimensional modal logic. *CoRR*, abs/1405.4100, 2014. arXiv:1405.4100.
- 34 Alexander M. Rabinovich and Boris A. Trakhtenbrot. Behaviour structures and nets. *Fundamenta Informaticae*, 11:357–404, 1988.
- 35 Wiesław Szwaś and Lidia Tendera. The guarded fragment with transitive guards. *Annals of Pure and Applied Logic*, 128(1):227–276, 2004. doi:10.1016/j.apal.2004.01.003.
- 36 P. S. Thiagarajan. Regular event structures and finite Petri nets: A conjecture. In W. Brauer, H. Ehrig, J. Karhumäki, and A. Salomaa, editors, *Formal and Natural Computing – Essays Dedicated to Grzegorz Rozenberg [on occasion of his 60th birthday]*, volume 2300 of *LNCS*, pages 244–256. Springer, 2002.
- 37 Robert J. van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37(4/5):229–327, 2001.
- 38 Walter Vogler. Deciding history preserving bisimilarity. In J. Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *Proceedings of ICALP'91*, volume 510 of *LNCS*, pages 495–505. Springer, 1991.
- 39 Hao Wang. Proving theorems by pattern recognition – I. *Communications of the ACM*, 3(4):220–234, 1960. doi:10.1145/367177.367224.
- 40 Hao Wang. Proving theorems by battern recognition – II. *The Bell System Technical Journal*, 40(1):1–41, 1961. doi:10.1002/j.1538-7305.1961.tb03975.x.
- 41 Glynn Winskel. Event Structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.
- 42 Glynn Winskel. Events, causality and symmetry. *Computer Journal*, 54(1):42–57, 2011.

Parameterized Complexity of Feature Selection for Categorical Data Clustering

Sayan Bandyapadhyay ✉ 


Department of Informatics, University of Bergen, Norway

Fedor V. Fomin ✉ 

Department of Informatics, University of Bergen, Norway

Petr A. Golovach ✉ 

Department of Informatics, University of Bergen, Norway

Kirill Simonov ✉ 

Algorithms and Complexity Group, TU Wien, Austria

Abstract

We develop new algorithmic methods with provable guarantees for feature selection in regard to categorical data clustering. While feature selection is one of the most common approaches to reduce dimensionality in practice, most of the known feature selection methods are heuristics. We study the following mathematical model. We assume that there are some inadvertent (or undesirable) features of the input data that unnecessarily increase the cost of clustering. Consequently, we want to select a subset of the original features from the data such that there is a small-cost clustering on the selected features. More precisely, for given integers ℓ (the number of irrelevant features) and k (the number of clusters), budget B , and a set of n categorical data points (represented by m -dimensional vectors whose elements belong to a finite set of values Σ), we want to select $m - \ell$ relevant features such that the cost of any optimal k -clustering on these features does not exceed B . Here the cost of a cluster is the sum of Hamming distances (ℓ_0 -distances) between the selected features of the elements of the cluster and its center. The clustering cost is the total sum of the costs of the clusters.

We use the framework of parameterized complexity to identify how the complexity of the problem depends on parameters k , B , and $|\Sigma|$. Our main result is an algorithm that solves the Feature Selection problem in time $f(k, B, |\Sigma|) \cdot m^{g(k, |\Sigma|)} \cdot n^2$ for some functions f and g . In other words, the problem is fixed-parameter tractable parameterized by B when $|\Sigma|$ and k are constants. Our algorithm for Feature Selection is based on a solution to a more general problem, Constrained Clustering with Outliers. In this problem, we want to delete a certain number of outliers such that the remaining points could be clustered around centers satisfying specific constraints. One interesting fact about Constrained Clustering with Outliers is that besides Feature Selection, it encompasses many other fundamental problems regarding categorical data such as Robust Clustering, Binary and Boolean Low-rank Matrix Approximation with Outliers, and Binary Robust Projective Clustering. Thus as a byproduct of our theorem, we obtain algorithms for all these problems. We also complement our algorithmic findings with complexity lower bounds.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms; Mathematics of computing \rightarrow Combinatorial algorithms

Keywords and phrases Robust clustering, PCA, Low rank approximation, Hypergraph enumeration

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.14

Related Version *Full Version:* <https://arxiv.org/abs/2105.03753>

Funding This work is supported by the Research Council of Norway via the project “MULTIVAL”.

Acknowledgements The authors are thankful to the anonymous reviewers for their helpful comments.



© Sayan Bandyapadhyay, Fedor V. Fomin, Petr A. Golovach, and Kirill Simonov; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 14; pp. 14:1–14:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Clustering is one of the most fundamental concepts in data mining and machine learning. A considerable challenge to the clustering approaches is the high dimensionality of modern datasets. When the data contains many irrelevant features (or attributes), an application of cluster analysis with a complete set of features could significantly decrease the solution's quality. The typical approach to overcome this challenge in practice is *feature selection*. The method is based on selecting a small subset of relevant features from the data and applying the clustering algorithm only on the selected features. The survey of [1] provides a comprehensive overview on methods for feature selection in clustering. Due to the significance of feature selection, there is a multitude of heuristic methods addressing the problem. However, very few provably correct methods are known [6, 7, 11].

Kim et al. [20] introduced a model of feature selection in the context of k -means clustering. We use their motivating example here. Decision-making based on market surveys is a pragmatic marketing strategy used by manufacturers to increase customer satisfaction. The respondents of a survey are segmented into similar-interest groups so that each group of customers can be treated in a similar way. Consider such a market survey data that typically contains responses of customers to a set of questions regarding their demographic and psychographic information, shopping experience, attitude towards new products and expectations from the business. The standard practice used by market managers to segment customers is to apply clustering techniques w.r.t. the whole set of features. However, depending on the application, responses corresponding to some of the features might not be relevant to find the target set of market segments. Also, some of the responses might contain incomplete or spurious information. To address this issue, Kim et al. [20] considered several quality criteria to return *Pareto* optimal (or non-dominated) solutions that optimize one or more criteria. One such solution removes a suitable subset of features and cluster the data w.r.t. the remaining features.

The main objective of this work is to study clustering problems on *categorical* data. In statistics, a categorical variable is a variable that can admit a fixed number of possible values. For example, it could be a gender, blood type, political orientation, etc. A prominent example of categorical data is binary data where the points are vectors each of whose coordinates can take value either 0 or 1. Binary data arise in several important applications. In electronic commerce, each transaction can be modeled as a binary vector (known as market basket data) each of whose coordinates denotes whether a particular item is purchased or not [32, 22]. The most common similarity (or dissimilarity) measure for categorical data objects is the Hamming distance, which is basically the number of mismatched attributes of the objects.

2 Our results

In this paper, we introduce a new model of feature selection w.r.t. categorical data clustering, which is motivated by the work of Kim et al. [20]. We assume that there are some inadvertent features of the input data that unnecessarily increase the cost of clustering. Consequently, in our model, we define the best subset of features (of a given size) as the subset that minimizes the corresponding cost of clustering. The goal is to compute such a subset and the respective clusters. We provide the first parameterized algorithmic and complexity results for feature selection in regard to categorical data clustering.

Let Σ be a finite set of non-negative integers. We refer to Σ as the alphabet and we denote the m -dimensional space over Σ by Σ^m . Given two m -dimensional vectors $\mathbf{x}, \mathbf{y} \in \Sigma^m$, the *Hamming distance* (or ℓ_0 -distance) $d_H(\mathbf{x}, \mathbf{y})$ is the number of different coordinates in \mathbf{x} and

\mathbf{y} , that is $d_H(\mathbf{x}, \mathbf{y}) = |\{i \in \{1, \dots, m\} : \mathbf{x}[i] \neq \mathbf{y}[i]\}|$. For a set of indices $S \subset \{1, 2, \dots, m\}$ and an $m \times n$ matrix \mathbf{A} , let \mathbf{A}^{-S} be the matrix obtained from \mathbf{A} by removing the rows with indices in S . We denote the columns of \mathbf{A}^{-S} by \mathbf{a}_{-S}^j for $1 \leq j \leq n$. We consider the following mathematical model of feature selection.

FEATURE SELECTION

Input: An alphabet Σ , an $m \times n$ matrix \mathbf{A} with columns $\mathbf{a}^1, \mathbf{a}^2, \dots, \mathbf{a}^n$ such that $\mathbf{a}^j \in \Sigma^m$ for all $1 \leq j \leq n$, a positive integer k , non-negative integers B and ℓ .

Task: Decide whether there is a subset $O \subset \{1, 2, \dots, m\}$ of size at most ℓ , a partition $\{I_1, I_2, \dots, I_k\}$ of $\{1, 2, \dots, n\}$, and vectors $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \Sigma^{m-|O|}$ such that

$$\sum_{i=1}^k \sum_{j \in I_i} d_H(\mathbf{a}_{-O}^j, \mathbf{c}_i) \leq B.$$

In the above definition and all subsequent problem definitions, without loss of generality, we assume that each cluster is non-empty, i.e., $I_i \neq \emptyset$ for each $1 \leq i \leq k$. Note that, otherwise, one could probe different values $k' < k$ for the actual number of non-empty clusters. The problem is defined as a decision problem, however if the instance is a yes-instance we would also like to find such a clustering. For $\ell = 0$, FEATURE SELECTION is the popular BINARY k -CLUSTERING problem, which is known to be NP-hard for every $k \geq 2$ [13]. This makes it natural to investigate the parameterized complexity of FEATURE SELECTION.

Our main result is the following theorem.

► **Theorem 1.** *FEATURE SELECTION is solvable in time $f(k, B, |\Sigma|) \cdot m^{g(k, |\Sigma|)} \cdot n^2$, where f and g are computable functions.*

In particular, this implies that for fixed k and $|\Sigma|$, the problem is fixed-parameter tractable (FPT¹) parameterized by B . Note that in any study concerning the parameterized complexity of a problem, the value of the parameter is implicitly assumed to be sufficiently small compared to the input size. Although the parameter B seems to be a natural choice from the problem definition, in general B can be fairly large. Hence, Theorem 1 is mostly applicable in the scenario when for the selected features the cost of clustering B is small and thus the points are well-clustered on the selected features. Even in this case the problem is far from being trivial. One can think of our problem as a problem from the broader class of editing problems, where the goal is to check whether a given instance is close to a “structured” one. In particular, our problem can be seen as the problem of editing the input matrix after removing at most ℓ rows such that the resulting matrix contains at most k distinct columns and the number of edits does not exceed the budget B . In this sense, our work is in line with the work of [18] on matrix completion and [15] on clustering. Note that in many applications it is reasonable to assume that k and $|\Sigma|$ are bounded, as the alphabet size and the number of clusters are small in practice. Indeed, for binary data, $|\Sigma| = 2$.

Another interesting property of our algorithm is that the running time does not depend on the number of irrelevant features ℓ . In particular, for fixed k , B , and $|\Sigma|$, it runs in polynomial time even when $\ell = \Omega(m)$. Also, the theorem could be used to identify the minimum number of irrelevant features ℓ such that the cost of k -clustering on the remaining features does not exceed B . Note that our time complexity also exponentially depends on the

¹ A problem is FPT or fixed-parameter tractable parameterized by a set of parameters if it can be solved by algorithms that are exponential only in the values of the parameters while polynomial in the size of the input.

14:4 Parameterized Complexity of Feature Selection for Categorical Data Clustering

number of clusters k . In this regard, one can compare our result with the result in [15] that shows that the binary version of the problem with $\ell = 0$ (BINARY k -CLUSTERING) is FPT parameterized by B only. However, in the presence of irrelevant features, the dependence on k is unavoidable as we state in our next theorem.

► **Theorem 2.** *FEATURE SELECTION is $W[1]$ -hard parameterized by*

- *either $k + (m - \ell)$*
- *or ℓ*

even when $B = 0$ and $\Sigma = \{0, 1\}$. Moreover, assuming the Exponential Time Hypothesis (ETH), the problem cannot be solved in time $f(k) \cdot m^{o(k)} \cdot n^{O(1)}$ for any function f , even when $B = 0$ and the alphabet Σ is binary.

Note that when $B = 0$ and $\Sigma = \{0, 1\}$, from Theorem 1 it follows that FEATURE SELECTION can be solved in time $f(k) \cdot m^{g(k)} \cdot n^2$. Theorem 2 shows that the dependence of such a function g on k is inevitable, unless $W[1] = \text{FPT}$, and $g(k)$ is unlikely to be sublinear up to ETH.

In order to prove Theorem 1, we prove a more general theorem about CONSTRAINED CLUSTERING WITH OUTLIERS. In this problem, one seeks a clustering with centers of clusters satisfying the property imposed by a set of relations. Constrained clustering [14] was introduced as the tool in the design of approximation algorithms for binary low-rank approximation problems. The CONSTRAINED CLUSTERING WITH OUTLIERS problem is basically the robust variant of this problem. As we will see, by the reduction given in Lemma 5, Theorem 4 proves Theorem 1. Besides FEATURE SELECTION, CONSTRAINED CLUSTERING WITH OUTLIERS encompasses a number of well-studied problems around robust clustering, low-rank matrix approximation, and dimensionality reduction. Our algorithm for constrained clustering implies fixed-parameter tractability for all these problems.

To define constrained clustering, we need a few definitions. A p -ary relation on Σ is a collection of p -tuples whose elements are in Σ .

► **Definition 3** (Vectors satisfying \mathcal{R}). *An ordered set $C = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_p\}$ of m -dimensional vectors in Σ^m is said to satisfy a set $\mathcal{R} = \{R_1, R_2, \dots, R_m\}$ of p -ary relations on Σ if for all $1 \leq i \leq m$, the p -tuple formed by the i -th coordinates of vectors from C , that is $(\mathbf{c}_1[i], \mathbf{c}_2[i], \dots, \mathbf{c}_p[i])$, belongs to R_i .*

We define the following constrained variant of robust categorical clustering.

CONSTRAINED CLUSTERING WITH OUTLIERS

Input: An alphabet Σ , an $m \times n$ matrix \mathbf{A} with columns $\mathbf{a}^1, \mathbf{a}^2, \dots, \mathbf{a}^n$ such that $\mathbf{a}^j \in \Sigma^m$ for all $1 \leq j \leq n$, a positive integer k , non-negative integers B and ℓ , a set $\mathcal{R} = \{R_1, R_2, \dots, R_m\}$ of k -ary relations on Σ .

Task: Decide whether there is a subset $O \subset \{1, 2, \dots, n\}$ of size at most ℓ , a partition $\mathcal{I} = \{I_1, I_2, \dots, I_k\}$ of $\{1, 2, \dots, n\} \setminus O$, and a set $C = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k\}$ of m -dimensional vectors in Σ^m such that C satisfies \mathcal{R} and

$$\sum_{i=1}^k \sum_{j \in I_i} d_H(\mathbf{a}^j, \mathbf{c}_i) \leq B.$$

Thus in CONSTRAINED CLUSTERING WITH OUTLIERS we want to identify a set of outliers $\mathbf{a}^i, i \in O$, such that the remaining $n - \ell$ vectors could be partitioned into k clusters $\{I_1, I_2, \dots, I_k\}$. Each cluster I_j could be identified by its center $\mathbf{c}_j \in \Sigma^m$ as the set of vectors that are closer to $\mathbf{c}_j \in \Sigma^m$ than to any other center (ties are broken arbitrarily). Then the

cost of each cluster I_j is the sum of the Hamming distances between its vectors and the corresponding center $\mathbf{c}_j \in \Sigma^m$. However, there is an additional condition that the set of cluster centers $C = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k\}$ must satisfy the set of k -ary relations \mathcal{R} . And, the total sum of costs of all clusters must not exceed B . We prove the following theorem.

► **Theorem 4.** *CONSTRAINED CLUSTERING WITH OUTLIERS is solvable in time*

$$(kB)^{O(kB)} |\Sigma|^{kB} \cdot n^{O(k)} \cdot m^2.$$

The connection between CONSTRAINED CLUSTERING WITH OUTLIERS and FEATURE SELECTION is established in the following lemma.

► **Lemma 5.** *For any instance (\mathbf{D}, k, B, ℓ) of FEATURE SELECTION, one can construct in time $\mathcal{O}(mn + k \cdot |\Sigma|^k)$ an equivalent instance $(\mathbf{A}, k', B', \ell', \mathcal{R})$ of CONSTRAINED CLUSTERING WITH OUTLIERS such that \mathbf{A} is the transpose of \mathbf{D} , $k' = |\Sigma|^k$, $B' = B$ and $\ell' = \ell$.*

Theorem 1 follows from Theorem 4 and Lemma 5. Connections of constrained clustering with several other clustering and low-rank matrix approximation problems have been established in the literature [14]. Similarly, Theorem 4 allows to design parameterized algorithms for robust variants of these problems.

Robust low-rank matrix approximation. Here we discuss two problems where for a given matrix of categorical data, we seek to remove ℓ columns such that the remaining columns are well approximated by a matrix of small rank. The vanilla case of the ℓ_0 -LOW RANK APPROXIMATION problem is the following. Given an $m \times n$ matrix \mathbf{A} over $\text{GF}(p)$ (a finite field of order p), the task is to find an $m \times n$ matrix \mathbf{B} over $\text{GF}(p)$ of $\text{GF}(p)$ -rank at most r which is closest to \mathbf{A} in the ℓ_0 -norm, i.e., the goal is to minimize $\|\mathbf{A} - \mathbf{B}\|_0$, the number of different entries in \mathbf{A} and \mathbf{B} . In the robust version of this problem, some of the columns of \mathbf{A} could be outliers, which brings us to the following problem.

ROBUST ℓ_0 -LOW RANK APPROXIMATION

Input: An $m \times n$ matrix \mathbf{A} over $\text{GF}(p)$, a positive integer r , non-negative integers B and ℓ .

Task: Decide whether there is a matrix \mathbf{B} of $\text{GF}(p)$ -rank at most r , and a matrix \mathbf{C} over $\text{GF}(p)$ with at most ℓ non-zero columns such that $\|\mathbf{A} - \mathbf{B} - \mathbf{C}\|_0 \leq B$.

Note that in this definition the non-zero columns of \mathbf{C} can take any values. However, it is easy to see that the problem would be equivalent if the columns of \mathbf{C} were constrained to be either zero columns or the respective columns of \mathbf{A} . This holds since if \mathbf{C} contains a non-zero column, it could be replaced by the respective column of \mathbf{A} , and the respective column of \mathbf{B} can be replaced by a zero column. This does not increase the cost nor the rank of \mathbf{B} . Thus any of the two formulations allows to restore the column outliers in \mathbf{A} from \mathbf{C} .

By a reduction [14, Lemma 1] similar to Lemma 5, we can show that Theorem 4 yields the following theorem.

► **Theorem 6.** *ROBUST ℓ_0 -LOW RANK APPROXIMATION is FPT parameterized by B when p and r are constants.*

Another popular variant of low-rank matrix approximation is the case when the approximation matrix \mathbf{B} is of low Boolean rank. More precisely, let \mathbf{A} be a binary $m \times n$ matrix. Now we consider the elements of \mathbf{A} to be *Boolean* variables. The *Boolean rank* of \mathbf{A} is the minimum r such that $\mathbf{A} = \mathbf{U} \wedge \mathbf{V}$ for a Boolean $m \times r$ matrix \mathbf{U} and a Boolean $r \times n$ matrix

\vee , where the product is Boolean, that is, the logical \wedge plays the role of multiplication and \vee the role of sum. The variant of the low Boolean-rank matrix approximation is the following problem.

ROBUST LOW BOOLEAN-RANK APPROXIMATION

Input: A binary $m \times n$ matrix \mathbf{A} , a positive integer r , non-negative integers B and ℓ .
Task: Decide whether there is a binary matrix \mathbf{B} of Boolean rank $\leq r$, and a binary matrix \mathbf{C} with at most ℓ non-zero columns such that $\|\mathbf{A} - \mathbf{B} - \mathbf{C}\|_0 \leq B$.

By Theorem 4 and reduction from constrained clustering to Boolean-rank matrix approximation identical to [14, Lemma 2], we have the following.

► **Theorem 7.** *ROBUST LOW BOOLEAN-RANK APPROXIMATION is FPT parameterized by B when r is a constant.*

Finally, we consider clustering with outliers. This problem looks very similar to feature selection. The only difference is that instead of features (the rows of the matrix \mathbf{A}), we seek to remove some columns of \mathbf{A} . More precisely, we consider the following problem.

k -CLUSTERING WITH COLUMN OUTLIERS

Input: An alphabet Σ , an $m \times n$ matrix \mathbf{A} with columns $\mathbf{a}^1, \mathbf{a}^2, \dots, \mathbf{a}^n$ such that $\mathbf{a}^j \in \Sigma^m$ for all $1 \leq j \leq n$, a positive integer k , non-negative integers B and ℓ .
Task: Decide whether there is a subset $O \subset \{1, 2, \dots, n\}$ of size at most ℓ , a partition of $\{1, 2, \dots, n\} \setminus O$ into k sets $\{I_1, I_2, \dots, I_k\}$ called clusters, and vectors $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in \Sigma^m$ such that the cost of clustering

$$\sum_{i=1}^k \sum_{j \in I_i} d_H(\mathbf{a}^j, \mathbf{c}_i) \leq B.$$

Note that k -CLUSTERING WITH COLUMN OUTLIERS is also a special case of CONSTRAINED CLUSTERING WITH OUTLIERS when every relation $R_i \in \mathcal{R}$ contains all possible k -tuples over Σ , that is, there are no constraints on the centers. Hence, by Theorem 4, we readily obtain the same result for this problem. However, in this special case we show that it is possible to obtain an improved result.

► **Theorem 8.** *k -CLUSTERING WITH COLUMN OUTLIERS is solvable in time $2^{O(B \log B)} |\Sigma|^B \cdot (nm)^{O(1)}$.*

In particular, the theorem implies that the problem is FPT parameterized by B and $|\Sigma|$. We note that the running time of Theorem 8 matches the running time in [15] obtained for the BINARY k -CLUSTERING problem without outliers on binary data. The interesting feature of the theorem is that the running time of the algorithm does not depend on the number of outliers ℓ , matching the bound of the problem without outliers. Most of the clustering procedures in robust statistics, data mining and machine learning perform well only for small number of outliers. Our theorem implies that if all of the inlier points could be naturally partitioned into k distinct clusters with small cost, then such a clustering could be efficiently recovered even after arbitrarily many outliers are added.

Related Work. CONSTRAINED CLUSTERING (without outliers) was introduced in [14] as a tool for designing EPTAS for LOW BOOLEAN-RANK APPROXIMATION. ROBUST ℓ_0 -LOW RANK APPROXIMATION is a variant of robust PCA for categorical data. The study of robust

PCA, where one seeks for a PCA when the input data is noisy or corrupted, is the large class of extensively studied problems, see the books [30, 8]. There are many models of robustness in the literature, most relevant to our work is the approach that became popular after the work of [9]. The variant of robust PCA where one seeks for identifying a set of outliers, also known as PCA with outliers, were studied in [4, 10, 31, 29].

For the vanilla variant, ℓ_0 -LOW RANK APPROXIMATION, a number of parameterized and approximation algorithms were developed [3, 14, 15, 21].

LOW BOOLEAN-RANK APPROXIMATION has attracted much attention, especially in the data mining and knowledge discovery communities. In data mining, matrix decompositions are often used to produce concise representations of data. Since much of the real data is binary or even Boolean in nature, Boolean low-rank approximation could provide a deeper insight into the semantics associated with the original matrix. There is a big body of work done on LOW BOOLEAN-RANK APPROXIMATION. We refer to [23, 25, 27, 26] for further references on this interesting problem. Parameterized algorithms for LOW BOOLEAN-RANK APPROXIMATION (without outliers) were studied in [15].

There are several approximations and parameterized algorithms known for BINARY k -CLUSTERING, which is the vanilla (without outliers) case of k -CLUSTERING WITH COLUMN OUTLIERS and with $\Sigma = \{0, 1\}$ [28, 14, 3, 16]. Most relevant to our work is the parameterized algorithm for BINARY k -CLUSTERING from [15]. Theorem 8 extends the result from [15] to clustering with outliers.

Paper Outline. In the remaining part of this extended abstract we focus on our algorithmic results. We briefly outline our techniques in Section 3. Then, in Section 4 we describe our main result, the FPT algorithm for CONSTRAINED CLUSTERING WITH OUTLIERS. Finally, in Section 5, we conclude with some open problems. Due to space constraints, the detailed presentation of the remaining results appears in the attached full version of the paper.

3 Our Techniques

Both of our algorithmic results, Theorems 4 and 8, have at their core the subhypergraph enumeration technique introduced by Marx [24]. This is fairly natural, since our algorithms solve generalized versions of the vanilla binary clustering problem, and the only known FPT algorithm [15] for the latter problem parameterized by B relies on the hypergraph enumeration as well. In fact, our algorithm for k -CLUSTERING WITH COLUMN OUTLIERS closely follows this established approach of applying the hypergraph construction to clustering problems ([16], and partly [15]). However, for the CONSTRAINED CLUSTERING WITH OUTLIERS problem the existing techniques do not work immediately. To deal with this, we generalize the previously used hypergraph construction. In what follows, we present the key ideas of both algorithms. We begin with the simpler case of k -CLUSTERING WITH COLUMN OUTLIERS, even though our main results are for FEATURE SELECTION and CONSTRAINED CLUSTERING WITH OUTLIERS.

For the presentation of our algorithms, we recall standard hypergraph notations and the notion of a fractional cover of a hypergraph. A hypergraph $G(V_G, E_G)$ consists of a set V_G of vertices and a set E_G of edges, where each edge is a subset of V_G . Consider two hypergraphs $H(V_H, E_H)$ and $G(V_G, E_G)$. We say that H appears in G at $V' \subseteq V_G$ as a partial hypergraph if there is a bijection π between V_H and V' such that for any edge $e \in E_H$, $\pi(e) \in E_G$, where $\pi(e) = \cup_{v \in e} \pi(v)$. H is said to appear in G at $V' \subseteq V_G$ as a subhypergraph if there is a bijection π between V' and V_H such that for any edge $e \in E_H$, there is an edge $e' \in E_G$ such that $\pi(e) = e' \cap V'$.

A fractional edge cover of H is an assignment $\phi : E_H \rightarrow [0, 1]$ such that for every vertex $v \in V_H$, the sum of the values assigned to the edges that contain v is at least 1, i.e., $\sum_{e \ni v} \phi(e) \geq 1$. The fractional cover number $\rho^*(H)$ of H is the minimum value $\sum_{e \in E} \phi(e)$ over all fractional edge covers ϕ of H . The following theorem is required for our algorithm.

► **Theorem 9** ([24]). *Let $H(V_H, E_H)$ be a hypergraph with fractional cover number $\rho^*(H)$, and let $G(V_G, E_G)$ be a hypergraph where each edge has size at most L . There is an algorithm that enumerates, in time $|V_H|^{O(|V_H|)} \cdot L^{|V_H| \rho^*(H) + 1} \cdot |E_G|^{\rho^*(H) + 1} \cdot |V_G|^2$, every subset $V \subseteq V_G$ where H appears in G as a subhypergraph.*

3.1 The Algorithm for k -Clustering with Column Outliers

Given an instance (\mathbf{A}, k, B, ℓ) of k -CLUSTERING WITH COLUMN OUTLIERS, we note that at most $2B$ distinct columns can belong to “nontrivial” clusters (with at least 2 distinct columns), exactly like in the case without the outliers. So we employ a color-coding scheme [2] to partition the columns in a way so that every column belonging to a nontrivial cluster of a fixed feasible solution is colored with its own color. Thus we reduce to multiple instances of the problem we call Restricted Clustering. In Restricted Clustering, we are given sets of columns U_1, U_2, \dots, U_p and a parameter B . The goal is to select p columns $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_p$ and a cluster center \mathbf{s} such that $\mathbf{b}_t \in U_t$ for $1 \leq t \leq p$ and $\sum_{i=1}^p d_H(\mathbf{b}_i, \mathbf{s}) \leq B$.

Restricted Clustering is similar to the Cluster Selection problem of [16] and [15], and the hypergraph-based algorithm to solve it is essentially the same as in [16]. However, next we briefly sketch the details, as this construction serves as the base for our more general CONSTRAINED CLUSTERING WITH OUTLIERS algorithm. First, guess $\mathbf{b}_1 \in U_1$ in the optimal solution to the instance of Restricted Clustering. If the cost of the optimal solution is at most B , then $d_H(\mathbf{b}_1, \mathbf{s})$ is at most B as well. If we know the set of at most B positions P where \mathbf{b}_1 and \mathbf{s} differ, we can easily identify \mathbf{s} by trying all possible $|\Sigma|^B$ options at these positions. For each option, we can find the closest \mathbf{b}_i from each U_i and check whether the total cost is at most B .

To show that we can enumerate all potential sets of deviating positions in FPT time, we identify the instance with the following hypergraph $H(V, E)$. The vertices V are the positions $\{1, \dots, m\}$. With each column \mathbf{x} in $\bigcup_{i=1}^p U_i$, we identify a hyperedge containing exactly the positions where \mathbf{x} is different from \mathbf{b}_1 . Now the optimal set of positions P and the optimal columns $\{\mathbf{b}_i\}_{i=1}^p$ induce a subhypergraph $H_0(V_0, E_0)$ of H such that $|V_0| \leq B$ and the fractional cover number of H_0 is at most two. The latter holds simply because wherever \mathbf{s} is different from \mathbf{x} , at least half of the chosen columns must also be different from \mathbf{x} , otherwise modifying \mathbf{s} to match \mathbf{x} decreases the cost. If we enumerate all possible subhypergraphs H_0 and all possible locations in H where they occur, we can surely find the optimal set of locations P . Since $|V_0| \leq B$, enumerating all choices for H_0 is clearly in FPT time. For a fixed H_0 , finding all occurrences in H is in FPT time by Theorem 9. Note that applying Theorem 9 results in FPT time only when the fractional cover number of H_0 is bounded by a constant. Also, by a sampling argument one can show that it suffices to consider only those H_0 with $O(\log B)$ hyperedges. It follows that the number of distinct hypergraphs that we have to consider for enumeration is bounded by only $2^{O(B \log B)}$. Thus it is possible to bound the dependence on B in the running time by $2^{O(B \log B)}$.

3.2 The Algorithm for Feature Selection

For FEATURE SELECTION, the above-mentioned approach is not applicable, for several reasons. Most crucially, it does not seem that one can partition the problem into k independent instances of a simpler single-center selection problem, in a way that we reduce k -CLUSTERING

WITH COLUMN OUTLIERS to k instances of Restricted Clustering (for a fixed coloring). Intuitively, the possibility to remove a subset of features does not allow such a partition as all the clusters depend simultaneously on the choice of those features. Moreover, our hardness result shows that for FEATURE SELECTION the running time cannot match with k -CLUSTERING WITH COLUMN OUTLIERS, as no $n^{o(k)}$ time algorithm is possible for constant B , assuming ETH.

By Lemma 5, for solving FEATURE SELECTION, it suffices to solve CONSTRAINED CLUSTERING WITH OUTLIERS. For the same reasons as with FEATURE SELECTION, the approach used for k -CLUSTERING WITH COLUMN OUTLIERS fails, as the constraints on the centers do not allow to form clusters independently. Instead, we generalize the hypergraph construction used for Restricted Clustering to handle the choice of all k centers simultaneously, as opposed to just one center at a time. This is the most technical part of the paper. The main idea is to base the hypergraph on k -tuples of columns instead of just single columns. In the next section, we formalize this intuitive discussion, presenting the proof in full detail.

4 The Algorithm for Constrained Clustering with Outliers

In this section we prove Theorem 4 by giving an algorithm for CONSTRAINED CLUSTERING WITH OUTLIERS that runs in $(kB)^{O(kB)} n^{O(k)} m^2 |\Sigma|^{kB}$ time. First, we prove a structural lemma that will be useful for analysis of the algorithm.

4.1 Structural Lemma

Here we show that an optimal set of centers corresponds to a “good” subhypergraph in a certain hypergraph. Consider a feasible clustering $\mathcal{I} = \{I_1, I_2, \dots, I_k\}$ having the minimum cost. Let $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k\}$ be a fixed set of centers corresponding to \mathcal{I} . Also, let T be the set of all tuples of the form $(\mathbf{a}^{i_1}, \mathbf{a}^{i_2}, \dots, \mathbf{a}^{i_k})$ such that $i_j \in I_j$ for all j . Note that we do not actually need to know the set T – we just introduce the notation for the sake of analysis. For a k -tuple $x = (\mathbf{x}_1, \dots, \mathbf{x}_k)$, we denote the tuple $(\mathbf{x}_1[j], \mathbf{x}_2[j], \dots, \mathbf{x}_k[j])$ by $x[j]$. Two k -tuples x and y are said to differ from each other at location j if $x[j] \neq y[j]$.

Let C be the k -tuple such that $C = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k)$. Suppose $x = (\mathbf{x}_1, \dots, \mathbf{x}_k)$ is such that there are at most B positions h where $x[h] \neq C[h]$, and for each $1 \leq j \leq m$, $x[j] \in R_j$. Consider the hypergraph H defined in the following way with respect to x . The labels of the vertices of H are in $\{1, 2, \dots, m\}$. For each k -tuple $y = (\mathbf{y}_1, \dots, \mathbf{y}_k)$ of T , we add an edge $S \subseteq \{1, 2, \dots, m\}$ such that $h \in S$ if $x[h] \neq y[h]$.

In the following lemma, we show that the hypergraph H has a “good” subhypergraph.

► **Lemma 10 (Structural Lemma).** *Suppose the input is a yes-instance. Consider a k -tuple $x = (\mathbf{x}_1, \dots, \mathbf{x}_k)$ such that there are at most B positions h where $x[h] \neq C[h]$ and for each $1 \leq j \leq m$, $x[j] \in R_j$. Also, consider the hypergraph H defined in the above with respect to x . There exists a subhypergraph $H^*(V^*, E^*)$ of H with the following properties.*

1. $|V^*| \leq B$.
2. $|E^*| \leq 200 \ln B$.
3. *The indices in V^* are the exact positions h such that $x[h] \neq C[h]$.*
4. *The fractional cover number of H^* is at most 4.*

To prove the above lemma, first, we show the existence of a subhypergraph that satisfies all the properties except the second one. Let P be the set of positions h such that $x[h] \neq C[h]$. Let $H_0(V_0, E_0)$ be the subhypergraph of H induced by P . By definition of x , P contains at most B indices. Thus, the first property follows immediately. The third property also follows, as $V_0 = P$, is exactly the set of positions $h \in \{1, 2, \dots, m\}$, where $x[h] \neq C[h]$. Next, we show that the fourth property holds for H_0 . In fact, we show a stronger bound.

► **Lemma 11.** *The fractional cover number of H_0 is at most 2.*

Proof. Note that the total number of edges of H_0 is $\tau = |T|$. We claim that each vertex of H_0 is contained in at least $\tau/2$ edges.

Consider any vertex h of H_0 . Suppose there is a $1 \leq j \leq k$, such that for at least $\lceil |I_j|/2 \rceil$ columns in I_j the value at location h is not $\mathbf{x}_j[h]$. Note that each such column contributes to $\prod_{t^1=1}^{j-1} |I_{t^1}| \cdot \prod_{t^2=j+1}^k |I_{t^2}| = \tau/|I_j|$ tuples $(\mathbf{y}_1, \dots, \mathbf{y}_k)$ of T such that $\mathbf{y}_j[h] \neq \mathbf{x}_j[h]$. Thus, the edge corresponding to each such tuple contains h . It follows that, at least $\lceil |I_j|/2 \rceil \cdot (\tau/|I_j|) \geq \tau/2$ edges in E_0 contain h .

In the other case, for all $1 \leq j \leq k$ and less than $\lceil |I_j|/2 \rceil$ columns in I_j , the value at location h is not $\mathbf{x}_j[h]$. We prove that this case does not occur. Note that there is a k -tuple z in R_h such that $z = x[h]$. Consider replacing $C[h]$ by z at position h of C . Next, we analyze the change in cost of the clustering \mathcal{I} . Note that the cost corresponding to positions other than h remains same. For a $1 \leq j \leq k$, if previously $\mathbf{c}_j[h] = \mathbf{x}_j[h]$, the cost remains same. Otherwise, $\mathbf{c}_j[h] \neq \mathbf{x}_j[h]$. Note that for more than $\lceil |I_j|/2 \rceil$ columns in I_j , the value at location h is $\mathbf{x}_j[h]$. Thus, by replacing $\mathbf{c}_j[h]$ by $\mathbf{x}_j[h]$, the cost decrement corresponding to the index j and location h is at least 1. As $x[h] \neq C[h]$, there is an index j such that $\mathbf{c}_j[h] \neq \mathbf{x}_j[h]$. It follows that the overall cost decrement is at least 1, which contradicts the optimality of the previously chosen centers. Hence, this case cannot occur. This completes the proof of the lemma. ◀

So far we have proved the existence of a subhypergraph that satisfies all the properties except the second. Next, we show the existence of a subhypergraph that satisfies all the properties. The following lemma completes the proof of Lemma 10. Its proof follows a standard sampling argument, and is presented in the full version.

► **Lemma 12.** *Let $B \geq 2$. Consider the subhypergraph H_0 that satisfies all the properties of Lemma 10 except (2). It is possible to select at most $200 \ln B$ edges from H_0 such that the subhypergraph H_0^* obtained by removing all the other edges from H_0 satisfies all the properties of Lemma 10.*

4.2 The Algorithm for Constrained Clustering

In this section, we describe our algorithm. The algorithm outputs a feasible clustering of minimum cost if there is a feasible clustering of the given instance. Otherwise, the algorithm returns “NO”.

The Algorithm. First, we consider all k -tuples $x = (\mathbf{x}_1, \dots, \mathbf{x}_k)$ such that \mathbf{x}_j is a column of \mathbf{A} for $1 \leq j \leq k$, and apply the following refinement process on each of them. Here, a k -tuple x of columns of \mathbf{A} is said to differ from \mathcal{R} at a position j for $1 \leq j \leq m$ if $x[j] \notin R_j$.

- Let $P \subseteq \{1, 2, \dots, m\}$ be the set of positions where x differs from \mathcal{R} .
- If $|P| > B$, probe the next k -tuple x .
- For each position $h \in P$, replace $x[h]$ by any element of R_h .

Next, for each refined k -tuple $x = (\mathbf{x}_1, \dots, \mathbf{x}_k)$, we construct a hypergraph G whose vertices are in $\{1, 2, \dots, m\}$. For each k -tuple $y = (\mathbf{y}_1, \dots, \mathbf{y}_k)$ of columns of \mathbf{A} , we add an edge $S \subseteq \{1, 2, \dots, m\}$ such that $h \in S$ if $x[h] \neq y[h]$. For all hypergraphs H_0^* having at most B vertices and at most $200 \ln B$ edges, we check if each vertex of H_0^* is contained in at least $1/4$ fraction of the edges. If that is the case, we use the algorithm of Theorem 9 to find every place P' where H_0^* appears in G as subhypergraph. For each such set P' , we perform

all possible $B' \leq B \cdot k$ edits of the tuple x at the locations in P' . In particular, for each B' , the editing is done in the following way. For each possible B' entries $(a_1, \dots, a_{B'})$ in $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ at the locations in P' and each set of B' symbols $(s_1, \dots, s_{B'})$ from Σ , we put s_j at the entry a_j for all j . After each such edit, we retrieve the edited tuple $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ and perform a sanity check on this tuple to ensure that it is a valid k -tuple center. In particular, for each index $1 \leq h \leq m$, if there is a $z \in R_h$ such that $z = x[h]$, we tag x as a valid tuple. Lastly, we output all such valid k -tuples as candidate centers if the corresponding cost of clustering is at most B . If no k -tuple is output as a candidate center, we return “NO”.

Note that, given a k -tuple candidate center $z = (z_1, \dots, z_k)$, one can compute a minimum cost clustering in the following greedy way, which implies that we can correctly compute the cost of clustering in the above. At each step i , we assign a new column of \mathbf{A} to a center. In particular, we add a column \mathbf{a}^j of \mathbf{A} to a cluster I'_t such that \mathbf{a}^j incurs the minimum cost over all unassigned columns if it is assigned to a center, i.e, it minimizes the quantity $\min_{t'=1}^k d(\mathbf{a}^j, z_{t'})$, and $z_{t'}$ is a corresponding center nearest to \mathbf{a}^j . As we are allowed to exclude ℓ outliers, we assign $n - \ell$ columns. The clustering $\{I'_1, \dots, I'_k\}$ obtained at the end of this process is the output. This finishes the description of our algorithm.

4.3 Analysis

Again consider the feasible clustering with partition $\mathcal{I} = \{I_1, I_2, \dots, I_k\}$ and the corresponding center tuple $C = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k)$ having the minimum cost. First, we have the following observation.

► **Observation 13.** *Suppose for a k -tuple x , x differs from C at B_1 positions. Then, after refinement, there is at most B_1 positions h such that $x[h] \neq C[h]$. Moreover, after refinement, $d_H(x, C) \leq B_1 \cdot k$.*

The first part is true for the following reason. If x was different from \mathcal{R} at a position h , then $x[h] \neq C[h]$ as well. Thus, refinement is applied for a position h where $x[h]$ already differs from $C[h]$. Hence, refinement does not affect a position h where $x[h] = C[h]$. The moreover part follows trivially from the first part as x is a k -tuple. Now, it is sufficient to prove the following lemma.

► **Lemma 14.** *Suppose there is a feasible clustering with partition $\mathcal{I} = \{I_1, I_2, \dots, I_k\}$ as defined above. The above algorithm successfully outputs the k -tuple $(\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k)$.*

Proof. Consider a k -tuple $x = (\mathbf{x}_1, \dots, \mathbf{x}_k)$ such that the column \mathbf{x}_j is in cluster I_j . As the algorithm considers all possible k -tuples of columns in \mathbf{A} , it must consider x . By Observation 13, after refinement, there are at most B positions h where $x[h] \neq C[h]$. Also, for each $1 \leq j \leq m$, $x[j] \in R_j$. Let G be the hypergraph constructed by the algorithm corresponding to this refined x . Note that the hypergraph H defined in Lemma 10 is a partial subhypergraph of G . Thus, the subhypergraph H^* of H is also a subhypergraph of G . As we enumerate all hypergraphs having at most B vertices, at most $200 \ln B$ edges and at most 4 fractional covering number, H^* must be considered by the algorithm. Let P' be the place in G where H^* appears. By the third property of Lemma 10, the locations in P' are the exact positions h such that $x[h] \neq C[h]$. It follows that an edit corresponding to P' generates the tuple $C = (\mathbf{c}_1, \dots, \mathbf{c}_k)$, as $d_H(x, C) \leq B \cdot k$. It is easy to see that C also passes the sanity check. Hence, C must be an output of the algorithm. ◀

Given the tuple center $C = (\mathbf{c}_1, \dots, \mathbf{c}_k)$, we use the greedy assignment scheme (described in the algorithm) to find the underlying clustering. Note that given any k -tuple candidate center $z = (z_1, \dots, z_k)$, this greedy scheme computes a minimum cost clustering with

z_1, \dots, z_k being the cluster centers. Thus, the cost of the clustering computed by the algorithm is at most B . Hence, the algorithm successfully outputs C as a candidate center. We summarize our findings in the following lemma.

► **Lemma 15.** *Suppose the input instance is a no-instance, then the algorithm successfully outputs “NO”. If the input instance is a yes-instance, the algorithm correctly computes a feasible clustering.*

4.4 Time Complexity

Next, we discuss the time complexity of our algorithm. The number of choices of x is $n^{O(k)}$. For each choice of x , the hypergraph G can be constructed in $n^{O(k)}$ time. The number of distinct hypergraphs H_0^* with at most B vertices and at most $200 \ln B$ edges is $2^{B \cdot 200 \ln B} = B^{O(B)}$, since there are 2^B possibilities for each edge. Now we analyze the time needed for locating a particular H_0^* in G . For any tuple $y \in T$, $d_H(y, C) \leq B$. By triangle inequality, $d_H(x, y) \leq 2B$. Thus, the size of any edge in H is at most $2B$, and we can remove any edge of G of size more than $2B$. From Theorem 9, it follows that every occurrence of H_0^* in G can be found in $B^{O(B)} \cdot (2B)^{4B+1} \cdot n^{4k+k} \cdot m^2 = B^{O(B)} \cdot n^{O(k)} m^2$ time. If H_0^* appears at some place in G , it would take $O((kB|\Sigma|)^{kB})$ time to edit x . Hence, in total the algorithm takes $(kB)^{O(kB)} |\Sigma|^B \cdot n^{O(k)} m^2$ time. By the above discussion, we have Theorem 4.

5 Conclusion

We initiated the systematic study of parameterized complexity of robust categorical data clustering problems. In particular, for k -CLUSTERING WITH COLUMN OUTLIERS, we proved that the problem can be solved in $2^{O(B \log B)} |\Sigma|^B \cdot (nm)^{O(1)}$ time. Further, we considered the case of row outliers and proved that FEATURE SELECTION is solvable in time $f(k, B, |\Sigma|) \cdot m^{g(k, |\Sigma|)} n^2$. We also proved that we cannot avoid the dependence on k in the degree of the polynomial of the input size in the running time unless $W[1] = \text{FPT}$, and the problem cannot be solved in $m^{o(k)} \cdot n^{O(1)}$ time, unless ETH is false. To deal with row outliers, we introduced the CONSTRAINED CLUSTERING WITH OUTLIERS problem and obtained the algorithm with running time $(kB)^{O(kB)} |\Sigma|^{kB} \cdot m^2 n^{O(k)}$. This problem is very general, and the algorithm for it not only allowed us to get the result for FEATURE SELECTION, but also led to the algorithms for the robust low rank approximation problems. In particular, we obtained that ROBUST ℓ_0 -LOW RANK APPROXIMATION is FPT if k and p are constants when the problem is parameterized by B . However, even if the low rank approximation problems are closely related to the matrix clustering problems, there are structural differences. For instance, we show that the complexity of clustering with column outliers is different from row outliers, however, low-rank approximation problems are symmetric. This leads to the question whether ROBUST ℓ_0 -LOW RANK APPROXIMATION, ROBUST LOW BOOLEAN-RANK APPROXIMATION and ROBUST PROJECTIVE CLUSTERING could be solved by better algorithms specially tailored for these problems. It is unlikely that potential improvements would considerably change the general qualitative picture. For example, ROBUST ℓ_0 -LOW RANK APPROXIMATION for $p = 2$ and $\ell = 0$ is NP-complete if $k = 2$ [12, 19] and W[1]-hard when parameterized by B [17]. It is also easy to observe that ROBUST ℓ_0 -LOW RANK APPROXIMATION for $p = 2$, $B = 0$ and $k = n - \ell - 1$ is equivalent to asking whether the input matrix A has $n - \ell$ linearly dependent columns. This immediately implies that ROBUST ℓ_0 -LOW RANK APPROXIMATION for $p = 2$ and $B = 0$ is W[1]-hard when parameterized by k or $n - \ell$ by the recent results about the EVEN SET problem [5]. The most interesting open question, by our opinion, is whether



the exponential dependence on k in the degree of the polynomial of the input size in the running time produced by our reduction of ROBUST ℓ_0 -LOW RANK APPROXIMATION to CONSTRAINED CLUSTERING WITH OUTLIERS could be avoided, even if p is a constant. Can the dependence of k be made polynomial (or even linear)?

References

- 1 Salem Alelyani, Jiliang Tang, and Huan Liu. Feature selection for clustering: A review. In Charu C. Aggarwal and Chandan K. Reddy, editors, *Data Clustering: Algorithms and Applications*, pages 30–373. CRC Press, 2013.
- 2 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
- 3 Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P. Woodruff. A PTAS for ℓ_p -low rank approximation. In *SODA'19*, pages 747–766. SIAM, 2019. doi:10.1137/1.9781611975482.47.
- 4 Aditya Bhaskara and Srivatsan Kumar. Low Rank Approximation in the Presence of Outliers. In *APPROX/RANDOM'18*, volume 116, pages 4:1–4:16, Dagstuhl, Germany, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.4.
- 5 Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *CoRR*, abs/1909.01986, 2019. arXiv:1909.01986.
- 6 Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. Unsupervised feature selection for the k -means clustering problem. In *NIPS'09*, pages 153–161. Curran Associates, Inc., 2009. URL: <http://papers.nips.cc/paper/3724-unsupervised-feature-selection-for-the-k-means-clustering-problem>.
- 7 Christos Boutsidis, Anastasios Zouzias, Michael W. Mahoney, and Petros Drineas. Randomized dimensionality reduction for k -means clustering. *IEEE Trans. Information Theory*, 61(2):1045–1062, 2015. doi:10.1109/TIT.2014.2375327.
- 8 Thierry Bouwmans, Necdet Serhat Aybat, and El-hadi Zahzah. *Handbook of robust low-rank and sparse matrix decomposition: Applications in image and video processing*. Chapman and Hall/CRC, 2016.
- 9 Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, 2011. doi:10.1145/1970392.1970395.
- 10 Yudong Chen, Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust matrix completion and corrupted columns. In *ICML'11*, pages 873–880, 2011.
- 11 Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k -means clustering and low rank approximation. In *STOC'15*, pages 163–172. ACM, 2015.
- 12 Chen Dan, Kristoffer Arnsfelt Hansen, He Jiang, Liwei Wang, and Yuchen Zhou. On low rank approximation of binary matrices. *CoRR*, abs/1511.01699, 2015. arXiv:1511.01699.
- 13 Uriel Feige. NP-hardness of hypercube 2-segmentation. *CoRR*, abs/1411.0821, 2014. arXiv:1411.0821.
- 14 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Approximation schemes for low-rank binary matrix approximation problems. *ACM Trans. Algorithms*, 16(1):12:1–12:39, 2020. doi:10.1145/3365653.
- 15 Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. Parameterized low-rank binary matrix approximation. *Data Min. Knowl. Discov.*, 34(2):478–532, 2020. doi:10.1007/s10618-019-00669-5.
- 16 Fedor V. Fomin, Petr A. Golovach, and Kirill Simonov. Parameterized k -Clustering: Tractability Island. In *FSTTCS'19*, volume 150 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.FSTTCS.2019.14.

- 17 Fedor V. Fomin, Daniel Lokshtanov, Syed Mohammad Meesum, Saket Saurabh, and Meirav Zehavi. Matrix rigidity from the viewpoint of parameterized complexity. *SIAM J. Discrete Math.*, 32(2):966–985, 2018. doi:10.1137/17M112258X.
- 18 Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. On the parameterized complexity of clustering incomplete data into subspaces of small rank. In *AAAI’20*, pages 3906–3913. AAAI Press, 2020.
- 19 Nicolas Gillis and Stephen A. Vavasis. On the complexity of robust PCA and ℓ_1 -norm low-rank matrix approximation. *CoRR*, abs/1509.09236, 2015. arXiv:1509.09236.
- 20 YongSeog Kim, W. Nick Street, and Filippo Menczer. Evolutionary model selection in unsupervised learning. *Intell. Data Anal.*, 6(6):531–556, 2002. URL: <http://content.iospress.com/articles/intelligent-data-analysis/ida00110>.
- 21 Ravi Kumar, Rina Panigrahy, Ali Rahimi, and David P. Woodruff. Faster algorithms for binary matrix factorization. In *ICML’19*, volume 97 of *Proceedings of Machine Learning Research*, pages 3551–3559. PMLR, 2019. URL: <http://proceedings.mlr.press/v97/kumar19a.html>.
- 22 Tao Li. A general model for clustering binary data. In *KDD’05*, pages 188–197, 2005.
- 23 Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, and Yuan Hong. Constraint-aware role mining via extended boolean matrix decomposition. *IEEE Trans. Dependable Sec. Comput.*, 9(5):655–669, 2012. doi:10.1109/TDSC.2012.21.
- 24 Dániel Marx. Closest substring problems with small distances. *SIAM J. Comput.*, 38(4):1382–1410, 2008. doi:10.1137/060673898.
- 25 Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE Trans. Knowl. Data Eng.*, 20(10):1348–1362, 2008. doi:10.1109/TKDE.2008.53.
- 26 Pauli Miettinen and Stefan Neumann. Recent developments in boolean matrix factorization. In *IJCAI’20*, pages 4922–4928. ijcai.org, 2020.
- 27 Pauli Miettinen and Jilles Vreeken. Model order selection for boolean matrix factorization. In *KDD’11*, pages 51–59. ACM, 2011. doi:10.1145/2020408.2020424.
- 28 Rafail Ostrovsky and Yuval Rabani. Polynomial-time approximation schemes for geometric min-sum median clustering. *J. ACM*, 49(2):139–156, 2002. doi:10.1145/506147.506149.
- 29 Kirill Simonov, Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. Refined complexity of PCA with outliers. In *ICML’19*, volume 97, pages 5818–5826. PMLR, 2019. URL: <http://proceedings.mlr.press/v97/simonov19a.html>.
- 30 René Vidal, Yi Ma, and S. Shankar Sastry. *Generalized Principal Component Analysis*, volume 40 of *Interdisciplinary applied mathematics*. Springer, 2016. doi:10.1007/978-0-387-87811-9.
- 31 Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust PCA via outlier pursuit. In *NIPS’10*, pages 2496–2504. Curran Associates, Inc., 2010. URL: <http://papers.nips.cc/paper/4005-robust-pca-via-outlier-pursuit>.
- 32 Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang. Binary matrix factorization with applications. In *ICDM’07*, pages 391–400. IEEE, 2007.

Decision Questions for Probabilistic Automata on Small Alphabets

Paul C. Bell  

Department of Computer Science, Liverpool John Moores University, UK

Pavel Semukhin  

Department of Computer Science, University of Oxford, UK

Abstract

We study the emptiness and λ -reachability problems for unary and binary Probabilistic Finite Automata (PFA) and characterise the complexity of these problems in terms of the degree of ambiguity of the automaton and the size of its alphabet. Our main result is that emptiness and λ -reachability are solvable in EXPTIME for polynomially ambiguous unary PFA and if, in addition, the transition matrix is over $\{0, 1\}$, we show they are in NP. In contrast to the Skolem-hardness of the λ -reachability and emptiness problems for exponentially ambiguous unary PFA, we show that these problems are NP-hard even for finitely ambiguous unary PFA. For binary polynomially ambiguous PFA with commuting transition matrices, we prove NP-hardness of the λ -reachability (dimension 9), nonstrict emptiness (dimension 37) and strict emptiness (dimension 40) problems.

2012 ACM Subject Classification Theory of computation \rightarrow Formal languages and automata theory; Theory of computation \rightarrow Computability; Theory of computation \rightarrow Probabilistic computation

Keywords and phrases Probabilistic finite automata, unary alphabet, emptiness problem, bounded ambiguity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.15

Related Version *Full Version:* <https://arxiv.org/abs/2105.10293>

1 Introduction

There are many possible extensions of the fundamental notion of a nondeterministic finite automaton. One such notion is that of a Probabilistic Finite Automata (PFA) which was first introduced by Rabin [19]. In a PFA \mathcal{P} over a (finite) input alphabet Σ the outgoing transitions from a state, for each input letter of Σ , form a probability distribution, as does the initial state vector. Thus, a word $w \in \Sigma^*$ is accepted with a certain probability, which we denote $\mathcal{P}(w)$.

There are a variety of interesting questions that one may ask about a PFA \mathcal{P} over an alphabet Σ . In this article we focus on two decision questions, that of λ -reachability and emptiness. The λ -reachability problem is stated thus: given a probability $\lambda \in [0, 1]$, does there exist some word $w \in \Sigma^*$ such that $\mathcal{P}(w) = \lambda$? In the (strict) emptiness problem, we ask if there exists *any* word $w \in \Sigma^*$ such that $\mathcal{P}(w) \geq \lambda$ (resp. $\mathcal{P}(w) > \lambda$). We also mention the related *cutpoint isolation* problem – to determine if for each $\epsilon > 0$, there exists a word $w \in \Sigma$ such that $|\mathcal{P}(w) - \lambda| < \epsilon$.

In general, the emptiness problem is undecidable for PFA [18], even over a binary alphabet when the automaton has 25 states [12]. The cutpoint isolation problem is undecidable [4] even for PFA with 420 states over a binary alphabet [5]. The problem is especially interesting given the seminal result of Rabin that if a cutpoint λ is isolated, then the cutpoint language associated with λ is necessarily regular [19].

We may ask which restrictions of PFA may lead to decidability of the previous problems. In this paper we are interested in PFA of *bounded ambiguity*, where the ambiguity of a word denotes the number of accepting runs of that word in the PFA. A PFA \mathcal{P} is f -ambiguous,



© Paul C. Bell and Pavel Semukhin;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 15; pp. 15:1–15:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for a function $f : \mathbb{N} \rightarrow \mathbb{N}$, if every word of length n has at most $f(n)$ accepting runs. A run is *accepting* if the probability of that run ending in a final state is strictly positive. The degree of ambiguity is thus a property of the NFA underlying a PFA (i.e. the NFA produced by setting all nonzero transition probabilities to 1). We may consider the notions of finite, polynomial or exponential ambiguity of \mathcal{P} based on whether f is bounded by a constant, is a polynomial or else is exponential, respectively. Characterisations of the degree of ambiguity of NFA are given by Weber and Seidel [23].

The authors of [8] show that emptiness for PFA remains undecidable even for polynomially ambiguous automata (quadratic ambiguity), show PSPACE-hardness results for finitely ambiguous PFA and that emptiness is in NP for the class of k -ambiguous PFA for every $k > 0$. The emptiness problem for PFA was later shown to be undecidable for linearly ambiguous automata [7].

Another restriction is to constrain input words of the PFA to come from a given language \mathcal{L} . If \mathcal{L} is a *letter-bounded* language, then the emptiness and λ -reachability problems remain undecidable for polynomially ambiguous PFA, even when all transition matrices commute [2]. In contrast, the *cutpoint-isolation* problem is decidable even for exponentially ambiguous PFA when inputs are constrained to come from a given letter-bounded context-free language, although it is NP-hard for 3-state PFA on letter-bounded inputs [3].

Our main results are as follows. We show that the λ -reachability and emptiness problems for probabilistic finite automata are:

- In EXPTIME for the class of polynomially ambiguous unary PFA and are NP-complete if, in addition, the transition matrix is over $\{0, 1\}$ [Theorem 4 and Corollary 11].
- NP-hard for polynomially ambiguous PFA over a binary alphabet with *fixed* and *commuting* transition matrices of dimension 40 (strict emptiness problem), 37 (nonstrict emptiness problem) and 9 (λ -reachability problem) [Theorem 12].

We also show NP-hardness for the class of finitely ambiguous unary PFA with $\{0, 1\}$ transition matrix [Theorem 10]. Our hardness results rely on the NP-hardness of solving binary quadratic equations and the universality problem for unary regular expressions. An interesting question, that is left open, is to find out the exact computational complexity of the above problems in the case of polynomially ambiguous unary PFA, i.e. to close the gap between the EXPTIME upper bound and NP lower bound.

2 Probabilistic Finite Automata and Notation

We denote by $\mathbb{Q}^{n \times n}$ the set of all $n \times n$ matrices over \mathbb{Q} . Given two column vectors $u \in \mathbb{Q}^n$ and $v \in \mathbb{Q}^m$, we denote by $[u|v]$ the column vector $(u_1, \dots, u_n, v_1, \dots, v_m)^T \in \mathbb{Q}^{n+m}$. For a sequence of vectors u_1, u_2, \dots, u_k , we write $[u_1|u_2|\dots|u_k]$ for the column vector which stacks the vectors on top of each other.

Given $A = (a_{ij}) \in \mathbb{Q}^{m \times m}$ and $B \in \mathbb{Q}^{n \times n}$, we define the direct sum $A \oplus B$ and Kronecker product $A \otimes B$ of A and B by:

$$A \oplus B = \left[\begin{array}{c|c} A & \mathbf{0}_{m,n} \\ \hline \mathbf{0}_{n,m} & B \end{array} \right], \quad A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & a_{22}B & \cdots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mm}B \end{bmatrix},$$

where $\mathbf{0}_{i,j}$ denotes the zero matrix of dimension $i \times j$. Note that neither \oplus nor \otimes are commutative in general. The following useful properties of \oplus and \otimes are well known.

► **Lemma 1.** *Let $A, B, C, D \in \mathbb{Q}^{n \times n}$. Then we have:*

- *Associativity: $(A \otimes B) \otimes C = A \otimes (B \otimes C)$ and $(A \oplus B) \oplus C = A \oplus (B \oplus C)$, thus $A \otimes B \otimes C$ and $A \oplus B \oplus C$ are unambiguous.*
- *Mixed product properties: $(A \otimes B)(C \otimes D) = (AC \otimes BD)$ and $(A \oplus B)(C \oplus D) = (AC \oplus BD)$.*
- *If A and B are stochastic matrices, then so are $A \oplus B$ and $A \otimes B$.*
- *If $A, B \in \mathbb{Q}^{n \times n}$ are both upper-triangular, then so are $A \oplus B$ and $A \otimes B$.*

See [13] for proofs of the first three properties of Lemma 1. The fourth property follows directly from the definition of the direct sum and Kronecker product and is not difficult to prove.

A Probabilistic Finite Automaton (PFA) \mathcal{P} with n states over an alphabet Σ is defined as $\mathcal{P} = (u, \{M_a | a \in \Sigma\}, v)$ where $u \in \mathbb{Q}^n$ is the *initial probability distribution*; $v \in \{0, 1\}^n$ is the *final state vector* and each $M_a \in \mathbb{Q}^{n \times n}$ is a (row) stochastic matrix. We will primarily be interested in *unary* and *binary* PFA, for which $|\Sigma| = 1$ and $|\Sigma| = 2$ respectively. For a word $w = a_1 a_2 \cdots a_k \in \Sigma^*$, we define the acceptance probability $\mathcal{P}(w) : \Sigma^* \rightarrow \mathbb{Q}$ of \mathcal{P} as:

$$\mathcal{P}(w) = u^T M_{a_1} M_{a_2} \cdots M_{a_k} v \in [0, 1],$$

which denotes the acceptance probability of w .¹

For a given cutpoint $\lambda \in [0, 1]$, we define the following languages: $L_{\geq \lambda}(\mathcal{P}) = \{w \in \Sigma^* \mid \mathcal{P}(w) \geq \lambda\}$, a nonstrict cutpoint language, and $L_{> \lambda}(\mathcal{P}) = \{w \in \Sigma^* \mid \mathcal{P}(w) > \lambda\}$, a strict cutpoint language. The (strict) emptiness problem for a cutpoint language is to determine if $L_{\geq \lambda}(\mathcal{P}) = \emptyset$ (resp. $L_{> \lambda}(\mathcal{P}) = \emptyset$). We are also interested in the λ -*reachability* problem, for which we ask if there exists a word $w \in \Sigma^*$ such that $\mathcal{P}(w) = \lambda$.

2.1 PFA Ambiguity

The degree of ambiguity of a finite automaton is a structural parameter, roughly indicating the number of accepting runs for a given input word. See [23] for a thorough discussion of ambiguity for nondeterministic automata and [2, 3, 7, 8] for connections to PFA.

Let $w \in \Sigma^*$ be an input word of an NFA $\mathcal{N} = (Q, \Sigma, \delta, Q_I, Q_F)$, with Q the set of states, Σ the input alphabet, $\delta \subset Q \times \Sigma \times Q$ the transition function, Q_I the set of initial states and Q_F the set of final states. For each $(p, w, q) \in Q \times \Sigma^* \times Q$, define $\text{da}_{\mathcal{N}}(p, w, q)$ as the number of paths for w in \mathcal{N} leading from state p to q . The *degree of ambiguity* of w in \mathcal{N} , denoted $\text{da}_{\mathcal{N}}(w)$, is defined as the number of all *accepting paths* for w (starting from an initial and ending in a final state). The *degree of ambiguity* of \mathcal{N} , denoted $\text{da}(\mathcal{N})$, is the supremum of the set $\{\text{da}_{\mathcal{N}}(w) \mid w \in \Sigma^*\}$. \mathcal{N} is called *infinitely ambiguous* if $\text{da}(\mathcal{N}) = \infty$, *finitely ambiguous* if $\text{da}(\mathcal{N}) < \infty$, and *unambiguous* if $\text{da}(\mathcal{N}) \leq 1$. The *degree of growth* of the ambiguity of \mathcal{N} , denoted $\text{deg}(\mathcal{N})$, is defined as the minimum degree of a univariate polynomial h with positive integral coefficients such that for all $w \in \Sigma^*$, $\text{da}_{\mathcal{N}}(w) \leq h(|w|)$ (if such a polynomial exists, in which case \mathcal{N} is called *polynomially ambiguous*, otherwise the degree of growth is infinite and \mathcal{N} is called *exponentially ambiguous*).

The above notions relate to NFA. We may derive an analogous notion of ambiguity for PFA by considering an embedding of a PFA \mathcal{P} to an NFA \mathcal{N} in such a way that for each letter $a \in \Sigma$, if the probability of transitioning from a state i to state j is nonzero under \mathcal{P} , then there is an edge from state i to j under \mathcal{N} for letter a . The initial states of \mathcal{N} are those of \mathcal{P} having nonzero initial probability and the final states of \mathcal{N} and \mathcal{P} coincide. We then say that \mathcal{P} is *finitely/polynomially/exponentially ambiguous* if \mathcal{N} is (respectively).

¹ Some authors interchange the order of u and v and use column stochastic matrices, although the two definitions are trivially equivalent.

15:4 Decision Questions for Probabilistic Automata on Small Alphabets

A state $q \in Q$ in an NFA (resp. PFA) is called *useful* if there exists an accepting path which visits q (resp. an accepting path of nonzero probability which visits q). We can characterise whether an NFA \mathcal{A} (and thus a PFA by the above embedding) has finite, polynomial or exponential ambiguity using the following properties:

EDA – There is a useful state $q \in Q$ such that, for some word $v \in \Sigma^*$, $da_{\mathcal{A}}(q, v, q) \geq 2$.

IDA_d – There are useful states $r_1, s_1, \dots, r_d, s_d \in Q$ and words $v_1, u_2, v_2, \dots, u_d, v_d \in \Sigma^*$ such that for all $1 \leq i \leq d$, r_i and s_i are distinct and $(r_i, v_i, r_i), (r_i, v_i, s_i), (s_i, v_i, s_i) \in \delta$ and for all $2 \leq i \leq d$, $(s_{i-1}, u_i, r_i) \in \delta$.

► **Theorem 2** ([14, 20, 23]). *An NFA (or PFA) \mathcal{A} having the EDA property is equivalent to it being exponentially ambiguous. For any $d \in \mathbb{N}$, an NFA (or PFA) \mathcal{A} having property IDA_d is equivalent to $\text{deg}(\mathcal{A}) \geq d$.*

Clearly, if \mathcal{N} agrees with IDA_d for some $d > 0$, then it also agrees with IDA₁, \dots , IDA_{d-1}. An NFA (or PFA) is thus finitely ambiguous if it does not possess property IDA₁.

3 Unary PFA

Our main focus is on unary automata. We begin by giving a simple folklore proof that the λ -reachability and emptiness problems are as computationally difficult as the famous Skolem problem, which is only known to be decidable for instances of depth 4 [22]. See also [1] for connections to reachability problems for Markov chains.

► **Theorem 3.** *The λ -reachability and emptiness problems for unary exponentially ambiguous Probabilistic Finite Automata are Skolem-hard.*

Proof. (*Folklore*). The λ -reachability problem for unary exponentially ambiguous PFA can be shown Skolem-hard based on the well known matrix formulation of Skolem's problem [11] and Turakainen's technique showing the equivalence of (strict) cutpoint language acceptance of generalised automata and exponentially ambiguous probabilistic automata [21].

The emptiness problem can be shown Skolem-hard by encoding the positivity problem which is known to be Skolem-hard, see [17] for example. ◀

We now move to prove our main result, specifically that the emptiness and λ -reachability problems for polynomially ambiguous unary probabilistic finite automata are in EXPTIME. Note again that without the restriction of polynomial ambiguity the problem is Skolem-hard by Theorem 3 and thus not even known to be decidable.

► **Theorem 4.** *The λ -reachability and (strict) emptiness problems for unary polynomially ambiguous Probabilistic Finite Automata are decidable in EXPTIME.*

In order to establish Theorem 4, we need to prove a series of lemmas.

The next lemma states that we may consider a unary polynomially ambiguous PFA whose transition matrix is upper-triangular. This will prove useful since in that case the eigenvalues of the transition matrix are rational nonnegative. In general, a polynomially ambiguous unary PFA may have a transition matrix with complex eigenvalues. The proof of the lemma relies on the analysis of strongly connected components (SCCs) of the underlying transition graph of a PFA.

► **Lemma 5.** Let $\mathcal{P} = (u, A, v)$ be a polynomially ambiguous unary Probabilistic Finite Automaton with acceptance function $\mathcal{P}(a^k) = u^T A^k v$. Then we can compute in EXPTIME a set of d polynomially ambiguous unary PFAs $\{\mathcal{P}_s = (u_s, U, v') \mid 0 \leq s \leq d-1\}$ such that U is rational upper-triangular and $\mathcal{P}(a^k) = \mathcal{P}_s(a^r) = u_s^T U^r v'$, where $k = rd + s$ with $0 \leq s \leq d-1$.

Proof. We will identify \mathcal{P} and its underlying graph in which an edge (p, q) exists iff $A_{p,q} \neq 0$. Two states p, q of a PFA are said to be *connected* if there exists a path from p to q and from q to p . We partition the set of states into Strongly Connected Components (SCC) denoted S_1, S_2, \dots, S_ℓ so that for any SCC S_j , either $|S_j| = 1$, or else any two states in S_j are connected. These SCCs can be computed in linear time.

A polynomially ambiguous PFA does not have the EDA property (see Sec. 2.1). This implies that every S_j , with $|S_j| > 1$, consists of a single directed cycle, possibly with transitions to other SCCs. To see this, suppose there are two different directed cycles inside S_j of lengths m and n and a common vertex p . Then one can construct two different paths of length mn from p to p by going m times along the first cycle and n time along the second cycle, respectively, contradicting the assumption that \mathcal{P} does not have the EDA property.

Note that if there exists a path from a state $p \in S_{j_1}$ to some $q \in S_{j_2}$, then there does not exist any path from any state in S_{j_2} to a state in S_{j_1} , otherwise S_{j_1} and S_{j_2} would merge to a single SCC (since all vertices are then connected). This implies that the connected components S_1, S_2, \dots, S_ℓ can be reordered in such a way that there are no transitions from S_j to S_i for $i < j$. Hence there exists a permutation matrix P such that the following matrix is stochastic block upper-triangular:

$$B = PAP^{-1} = \begin{pmatrix} B_1 & * & \cdots & * \\ 0 & B_2 & \ddots & * \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & B_\ell \end{pmatrix},$$

such that each $B_j \in \mathbb{Q}^{d_j \times d_j}$, where d_j is the size of S_j , and $B_j \preceq P_j$, where $P_j \in \mathbb{N}^{d_j \times d_j}$ is a permutation matrix, and the entries $*$ are arbitrary. Here $M \preceq N$ means that M is entrywise less than N , i.e. $M_{i,j} \leq N_{i,j}$.

Let $d = \text{lcm}\{d_j \mid 1 \leq j \leq \ell\}$ (in fact, we can simply take $d = \prod_{j=1}^{\ell} d_j$). We then see that:

$$U := B^d = PA^d P^{-1} = \begin{pmatrix} B_1^d & * & \cdots & * \\ 0 & B_2^d & \ddots & * \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & B_\ell^d \end{pmatrix}.$$

Note that each $B_j^d \preceq P_j^d = I_j$, where $I_j \in \mathbb{N}^{d_j \times d_j}$ is the identity matrix, and the entries $*$ are arbitrary. Therefore, each B_j^d is diagonal, and so U is clearly upper-triangular.

We then define $\mathcal{P}_s = (u_s, U, v')$, for $0 \leq s \leq d-1$, with $u_s^T = u^T A^s P^{-1}$ and $v' = Pv$ noting that Pv is a binary vector as required of a final state vector. We now see that:

$$\mathcal{P}(a^k) = u^T A^k v = u^T A^s A^{rd} v = u^T A^s P^{-1} (PA^{rd} P^{-1}) Pv = u_s^T U^r v' = \mathcal{P}_s(a^r)$$

for $k = rd + s$ with $0 \leq s \leq d-1$ as required. Here we used the identity $U^r = PA^{rd} P^{-1}$.

Finally, note that d can be exponential in the number of states of \mathcal{P} , which in turn is bounded by the input size. Hence computing U and all u_s , for $0 \leq s \leq d-1$, takes exponential time. ◀

15:6 Decision Questions for Probabilistic Automata on Small Alphabets

The next lemma gives us an efficient method to compute an explicit formula for the acceptance probability function of a unary PFA with upper-triangular transition matrix.

► **Lemma 6.** *Let $\mathcal{P} = (u, A, v)$ be a unary probabilistic finite automaton such that A is rational upper-triangular, and let $\lambda_0 = 1 > \lambda_1 > \dots > \lambda_m \geq 0$ be distinct eigenvalues of A . Then there exist a constant $c \in \mathbb{Q}$ and univariate polynomials p_1, \dots, p_m over \mathbb{Q} , all of which can be computed in polynomial time, such that*

$$\mathcal{P}(a^k) = c + \sum_{i=1}^m p_i(k) \lambda_i^k.$$

Proof. First, we write A in *Jordan normal form* $A = S^{-1}JS$, where S is a nonsingular ($\det(S) \neq 0$) matrix consisting of the generalised eigenvectors of A . Recall that A is a rational upper-triangular matrix. It follows that J and S must have rational entries. Moreover, to compute J and S , we need to solve systems of linear equations over \mathbb{Q} , which can be done in polynomial time. Computing S^{-1} also requires polynomial time. Matrix J has the form $J = \bigoplus_{i=0}^m \bigoplus_{j=1}^{n_i} J_{\ell_{i,j}}(\lambda_i)$, where $J_{\ell_{i,j}}(\lambda_i)$ is a $\ell_{i,j} \times \ell_{i,j}$ *Jordan block* and n_i is the geometric multiplicity of λ_i (hence $\sum_{j=1}^{n_i} \ell_{i,j}$ is the algebraic multiplicity of λ_i). Recall that a Jordan block $J_\ell(\lambda)$ of size $\ell \times \ell$ that corresponds to an eigenvalue λ has the form:

$$J_\ell(\lambda) = \begin{pmatrix} \lambda & 1 & 0 & \dots & 0 \\ 0 & \lambda & 1 & \dots & 0 \\ 0 & 0 & \lambda & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda \end{pmatrix} \in \mathbb{Q}^{\ell \times \ell}.$$

Noting that $\binom{x}{y} = 0$ if $y > x$, we see that

$$J_\ell(\lambda)^k = \begin{pmatrix} \lambda^k & \binom{k}{1} \lambda^{k-1} & \binom{k}{2} \lambda^{k-2} & \dots & \binom{k}{\ell-1} \lambda^{k-(\ell-1)} \\ 0 & \lambda^k & \binom{k}{1} \lambda^{k-1} & \dots & \binom{k}{\ell-2} \lambda^{k-(\ell-2)} \\ 0 & 0 & \lambda^k & \dots & \binom{k}{\ell-3} \lambda^{k-(\ell-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda^k \end{pmatrix}. \quad (1)$$

Note that the entries of $J_\ell(\lambda)^k$ have the form $q_{i,j}(k) \lambda^k$, where $q_{i,j}(k)$ are polynomials over \mathbb{Q} that can be computed in polynomial time. Namely, $q_{i,i+p}(k) = \binom{k}{p} \lambda^{-p}$ for $0 \leq p \leq \ell - i$, and $q_{i,j}(k) = 0$ for $i > j$. Note that even though p appears in the exponent of λ^{-p} and as $p!$ in $\binom{k}{p}$, these values are still computable in PTIME from the input data because p is bounded by the dimension of the matrix, which in turn is bounded by the input size.

Next, we note that $J^k = \bigoplus_{i=0}^m \bigoplus_{j=1}^{n_i} J_{\ell_{i,j}}(\lambda_i)^k$. Hence the entries of J^k have the form $p_{s,t}(k) \lambda_i^k$, where $p_{s,t}(k)$ are polynomials over \mathbb{Q} . So we can write the function $\mathcal{P}(a^k)$ as follows:

$$\mathcal{P}(a^k) = u^T A^k v = (u^T S^{-1}) J^k (Sv).$$

Note that in the above equation, $u^T S^{-1}$ and Sv are rational vectors. It follows that

$$\mathcal{P}(a^k) = \sum_{i=0}^m p_i(k) \lambda_i^k$$

for some polynomials $p_i(k)$ over \mathbb{Q} . In fact, these polynomials are rational linear combinations of those $p_{s,t}(k)$ that multiply λ_i^k in the expression for J^k , and so they can be computed in polynomial time.

Finally, recall that $\lambda_0 = 1$ and note that the Jordan blocks that correspond to the dominant eigenvalues of a stochastic matrix have size 1×1 (for the proof of this fact see, e.g. [9, Theorem 6.5.3]). It follows from (1) that the terms λ_0^k in the formula for J^k are multiplied by constant polynomials $p_{s,t}(k) = 1$. Hence $p_0(k) = c$ for some constant $c \in \mathbb{Q}$. ◀

The next technical lemma is crucial in our later analysis of the running time of the algorithms for the emptiness and λ -reachability problems presented in Lemmas 8 and 9.

► **Lemma 7.** *Let $D \in \mathbb{R}$ be such that $\ln D > 2$. Then for all $x > 3D \ln D$, we have $D \ln x < x$.*

Proof. Our goal is to find $x_0 > 0$ such that every $x > x_0$ satisfies $D \ln x < x$. First, let us make a substitution $x = Dt$, where $t > 1$. Then we can rewrite $D \ln x < x$ as follows

$$\begin{aligned} D \ln(Dt) &< Dt, \\ \ln t + \ln D &< t. \end{aligned}$$

We want to find $t_0 > 1$ such that every $t > t_0$ satisfies $\ln t + \ln D < t$. Let us make another substitution $t = \ln D + u \ln \ln D$, where $u > 0$. Then we can write the previous inequality as

$$\begin{aligned} \ln(\ln D + u \ln \ln D) + \ln D &< \ln D + u \ln \ln D, \\ \ln \left(\ln D \left(1 + u \frac{\ln \ln D}{\ln D} \right) \right) &< u \ln \ln D, \\ \ln \ln D + \ln \left(1 + u \frac{\ln \ln D}{\ln D} \right) &< u \ln \ln D. \end{aligned} \tag{2}$$

So we need to find $u_0 > 0$ such that for all $u > u_0$, the inequality (2) holds. In order to do this, we can replace the left-hand side of (2) with a larger value using $\ln \left(1 + u \frac{\ln \ln R}{\ln R} \right) < u \frac{\ln \ln R}{\ln R}$. Thus we obtain

$$\begin{aligned} \ln \ln D + u \frac{\ln \ln D}{\ln D} &< u \ln \ln D, \\ 1 + \frac{u}{\ln D} &< u, \quad \ln D + u < u \ln D, \quad \frac{\ln D}{\ln D - 1} < u. \end{aligned}$$

Recall that by our assumption $\ln D > 2$. In this case, $\frac{\ln D}{\ln D - 1} < 2$, and hence we can choose $u_0 = 2$. This gives us the values $t_0 = \ln D + u_0 \ln \ln D = \ln D + 2 \ln \ln D$ and $x_0 = Dt_0 = D(\ln D + 2 \ln \ln D)$. Since $\ln \ln D < \ln D$, we can choose x_0 to be $x_0 = 3D \ln D$. ◀

We now proceed to the proof of our main result. We split the analysis into two cases depending on whether or not the cutpoint λ coincides with the limit $\lim_{k \rightarrow \infty} \mathcal{P}(a^k)$, which is unique by Lemma 6.

► **Lemma 8.** *Let $\mathcal{P} = (u, A, v)$ be a unary probabilistic finite automaton such that A is rational upper-triangular, and let $\lambda \in [0, 1] \cap \mathbb{Q}$ be a cutpoint. Assuming that $\lambda \neq \lim_{k \rightarrow \infty} \mathcal{P}(a^k)$, the (strict) emptiness and λ -reachability problems for \mathcal{P} and λ are decidable in EXPTIME.*

Proof. By Lemma 6, we can write $\mathcal{P}(a^k) = c + \sum_{i=1}^m p_i(k) \lambda_i^k$, where $1 > \lambda_1 > \dots > \lambda_m$ are the eigenvalues of A and c and the coefficients of p_i are rational numbers that can be computed in polynomial time. By assumption, $\lim_{k \rightarrow \infty} \mathcal{P}(a^k) = c \neq \lambda$. Let $\epsilon = \frac{|c-\lambda|}{2}$. We now determine a natural number $k_0 > 0$ such that $\mathcal{P}(a^k) \in (c - \epsilon, c + \epsilon)$ for all $k > k_0$.

15:8 Decision Questions for Probabilistic Automata on Small Alphabets

Let each $p_i(k)$ have the form $p_i(k) = a_{i,s}k^s + a_{i,s-1}k^{s-1} + \dots + a_{i,0}$, where $s \leq n$ is the size of the largest Jordan block in the Jordan normal form of A (we do not assume here that $a_{i,s} \neq 0$). Then for all $k > 0$ we have

$$\left| \sum_{i=1}^m p_i(k) \lambda_i^k \right| \leq \lambda_1^k \sum_{i=1}^m |p_i(k)| \leq \lambda_1^k k^s \sum_{i=1}^m \sum_{j=0}^s |a_{i,j}| = d k^s \lambda_1^k,$$

where $d = \sum_{i=1}^m \sum_{j=0}^s |a_{i,j}| \in \mathbb{Q}$ can be computed in polynomial time by Lemma 6.

Let $k_1 > 0$ be a number to be defined later such that for all $k > k_1$,

$$k^s < \left(\frac{1}{\sqrt{\lambda_1}} \right)^k = \lambda_1^{-\frac{k}{2}}.$$

Then for all $k > k_1$, we have $d k^s \lambda_1^k < d \lambda_1^{\frac{k}{2}}$. Thus we need to find $k_0 \geq k_1$ such that for all $k > k_0$, we have $\lambda_1^{\frac{k}{2}} < \epsilon/d$. Note that if $\epsilon/d \geq 1$, then we can take $k_0 = k_1$. Hence we assume that $\epsilon/d < 1$.

The inequality $\lambda_1^{\frac{k}{2}} < \epsilon/d$ is equivalent to $k \ln \lambda_1 < 2 \ln(\epsilon/d)$. Since $\ln \lambda_1 < 0$, the previous inequality is equivalent to

$$k > \frac{2 \ln(\epsilon/d)}{\ln \lambda_1} = \frac{2 \ln(d/\epsilon)}{-\ln \lambda_1}. \quad (3)$$

To determine k_0 , we need an upper bound on the right-hand side of (3). We will use the fact that for any rational $r > 1$, $\ln r < \log_2 r \leq \log_2 \lceil r \rceil < \text{bins}(\lceil r \rceil)$, where $\text{bins}(n)$ is the size of the binary representation of n . Thus $\text{bins}(\lceil r \rceil)$ gives a polynomially computable integer upper bound for $\ln r$.

Next, using the fact that $\ln(1+x) < x$ for $x \neq 0$, we obtain

$$\ln \lambda_1 = \ln(1 + (\lambda_1 - 1)) < \lambda_1 - 1,$$

which gives $-\ln \lambda_1 > 1 - \lambda_1$. Hence a polynomially computable upper bound on the right-hand side of (3) is

$$\frac{2 \ln(d/\epsilon)}{-\ln \lambda_1} < \frac{2 \text{bins}(\lceil d/\epsilon \rceil)}{1 - \lambda_1}. \quad (4)$$

Next we compute a value k_1 such that for all $k > k_1$:

$$k^s < \lambda_1^{-\frac{k}{2}} \quad \text{or, equivalently,} \quad C \ln k < k, \quad (5)$$

where $C = \frac{2s}{-\ln \lambda_1}$. Using the fact that $\ln(1+x) < x$ for $x \neq 0$, we obtain $C < \frac{2s}{1 - \lambda_1}$.

Hence in order to find k_1 , we can replace C in (5) with $D = \frac{2s}{1 - \lambda_1}$. In addition, we can assume that $\ln D > 2$, since otherwise we can replace D with a larger value that satisfies this condition, e.g. with $D = 9$. Now, Lemma 7 implies that every $k > 3D \ln D$ satisfies $D \ln k < k$. To make this value polynomially computable, we can choose it to be

$$k_1 = 3 \lceil D \rceil \text{bins}(\lceil D \rceil), \quad \text{where } D = \max \left\{ \frac{2s}{1 - \lambda_1}, 9 \right\}.$$

Finally, combining the right-hand side of (4) with the above formula, we can define

$$k_0 = \max \left\{ \frac{2 \text{bins}(\lceil d/\epsilon \rceil)}{1 - \lambda_1}, 3 \lceil D \rceil \text{bins}(\lceil D \rceil) \right\}.$$

Note that all the values that appear in the above formula, e.g. ϵ , d and D , can be computed in polynomial time from the input data.

At this point we have derived a polynomially computable k_0 such that $\mathcal{P}(a^k) = u^T A^k v \in (c - \epsilon, c + \epsilon)$ and, in particular, $\mathcal{P}(a^k) \neq \lambda$ for all $k > k_0$. Now, to decide the λ -reachability problem, we need to check for each integer $k \in [0, k_0]$ whether $u^T A^k v = \lambda$. Note that the number of integers in $[0, k_0]$ is equal to $2^{\text{bins}(k_0)}$, which is exponential in the instance size. Also, computing A^k for a given $k \in [0, k_0]$ takes exponential time because $\text{bins}(A^k) = \mathcal{O}(2^{\text{bins}(k_0)} \text{bins}(A))$. So, the overall algorithm is in EXPTIME.

In a similar way, we can decide the (strict) emptiness problem in EXPTIME. For instance, suppose $\lambda > c$. Then for all $k > k_0$, we have $\mathcal{P}(a^k) < c + \epsilon < \lambda$. Thus deciding whether there exists k such that $\mathcal{P}(a^k) < \lambda$ is trivial. Suppose we want to know if there exists k such that $\mathcal{P}(a^k) \geq \lambda$. In this case, we need to check for each integer $k \in [0, k_0]$ whether $u^T A^k v \geq \lambda$. By the same argument as before, this can be done in EXPTIME. \blacktriangleleft

► Lemma 9. *Let $\mathcal{P} = (u, A, v)$ be a unary polynomially ambiguous probabilistic finite automaton such that A is upper-triangular and let $\lambda \in [0, 1] \cap \mathbb{Q}$ be a cutpoint. Assuming that $\lambda = \lim_{k \rightarrow \infty} \mathcal{P}(a^k)$, the (strict) emptiness and λ -reachability problems for \mathcal{P} and λ are decidable in EXPTIME.*

Proof. Recall that by Lemma 6, we can write $\mathcal{P}(a^k) = c + \sum_{i=1}^m p_i(k) \lambda_i^k$, where $1 > \lambda_1 > \dots > \lambda_m$ are the eigenvalues of A and c and the coefficients of p_i are rational numbers computable in polynomial time. By our assumption, $\lambda = \lim_{k \rightarrow \infty} \mathcal{P}(a^k) = c$. As before, let each $p_i(k)$ have the form $p_i(k) = a_{i,s} k^s + a_{i,s-1} k^{s-1} + \dots + a_{i,0}$, where $s \leq n$ (we do not assume here that $a_{i,s} \neq 0$).

In addition, assume that the leading coefficient of $p_1(k)$ is $a_{1,t}$, for some $t \leq s$. Without loss of generality, suppose $a_{1,t} > 0$; the case when $a_{1,t} < 0$ is similar. First, we compute k_0 such that $p_1(k) > \frac{1}{2} a_{1,t} k^t$ for all $k > k_0$. To do this, we will use the following inequalities:

$$a_{1,t} k^t + a_{1,t-1} k^{t-1} + \dots + a_{1,0} > \frac{1}{2} a_{1,t} k^t \iff \frac{1}{2} a_{1,t} k^t + a_{1,t-1} k^{t-1} + \dots + a_{1,0} > 0$$

$$\text{and } |a_{1,t-1} k^{t-1} + \dots + a_{1,0}| \leq k^{t-1} (|a_{1,t-1}| + \dots + |a_{1,0}|) = k^{t-1} \sum_{j=0}^{t-1} |a_{1,j}| \quad \text{if } k \geq 1.$$

So, the inequality $p_1(k) > \frac{1}{2} a_{1,t} k^t$ follows from $\frac{1}{2} a_{1,t} k^t > k^{t-1} \sum_{j=0}^{t-1} |a_{1,j}|$, which is equivalent to $k > \frac{2}{a_{1,t}} \sum_{j=0}^{t-1} |a_{1,j}|$. Therefore, we conclude that

$$p_1(k) > \frac{1}{2} a_{1,t} k^t \quad \text{for all } k \text{ such that } k > k_0 := \max \left\{ 1, \frac{2}{a_{1,t}} \sum_{j=0}^{t-1} |a_{1,j}| \right\}. \quad (6)$$

Now we want to find $k_1 \geq k_0$ such that for all $k > k_1$, we have

$$\lambda_1^k p_1(k) + \lambda_2^k p_2(k) + \dots + \lambda_m^k p_m(k) > 0. \quad (7)$$

Note that

$$|\lambda_2^k p_2(k) + \dots + \lambda_m^k p_m(k)| \leq \lambda_2^k (|p_2(k)| + \dots + |p_m(k)|) \leq d k^s \lambda_2^k, \quad (8)$$

where $d = \sum_{i=2}^m \sum_{j=0}^s |a_{i,j}|$. Using (6) and (8), we see that (7) holds whenever $k > k_0$ and $d k^s \lambda_2^k < \frac{1}{2} a_{1,t} k^t \lambda_1^k$, which is equivalent to

15:10 Decision Questions for Probabilistic Automata on Small Alphabets

$$\frac{2dk^{s-t}}{a_{1,t}} < \left(\frac{\lambda_1}{\lambda_2}\right)^k \quad \text{or} \quad \ln \frac{2d}{a_{1,t}} + (s-t) \ln k < k \ln \frac{\lambda_1}{\lambda_2}$$

$$\frac{1}{\ln \lambda_1/\lambda_2} \left(\ln \frac{2d}{a_{1,t}} + (s-t) \ln k \right) < k. \quad (9)$$

We will use the following inequality

$$\ln \frac{\lambda_1}{\lambda_2} = -\ln \frac{\lambda_2}{\lambda_1} = -\ln \left(1 + \frac{\lambda_2 - \lambda_1}{\lambda_1} \right) > -\frac{\lambda_2 - \lambda_1}{\lambda_1} > \lambda_1 - \lambda_2.$$

Then we can replace (9) with a stronger inequality

$$\frac{1}{\lambda_1 - \lambda_2} \left(\ln \frac{2d}{a_{1,t}} + (s-t) \ln k \right) < k. \quad (10)$$

In the following, we will assume $t < s$ since otherwise (10) simplifies to $\frac{1}{\lambda_1 - \lambda_2} \ln \frac{2d}{a_{1,t}} < k$.

Let us make the substitution $k = t \left(\frac{2d}{a_{1,t}} \right)^{-\frac{1}{s-t}}$, where $t > 0$. Then (10) can be written as

$$\frac{1}{\lambda_1 - \lambda_2} \left(\ln \frac{2d}{a_{1,t}} + (s-t) \ln t + (s-t) \frac{-1}{s-t} \ln \frac{2d}{a_{1,t}} \right) < t \left(\frac{2d}{a_{1,t}} \right)^{-\frac{1}{s-t}}$$

$$\left(\frac{2d}{a_{1,t}} \right)^{\frac{1}{s-t}} \frac{s-t}{\lambda_1 - \lambda_2} \ln t < t.$$

Let $D = \max \left\{ 9, \left(\frac{2d}{a_{1,t}} \right)^{\frac{1}{s-t}} \frac{s-t}{\lambda_1 - \lambda_2} \right\}$. Here 9 is needed to satisfy the requirement $\ln D > 2$ in Lemma 7. Then by Lemma 7, the above inequality holds when $t > 3D \ln D$. Therefore, (10) and hence (9) holds when $k > 3 \left(\frac{2d}{a_{1,t}} \right)^{-\frac{1}{s-t}} D \ln D$. To make this bound polynomially computable, we can simplify it as follows. Suppose that $2d \geq a_{1,t}$. Then (9) holds when

$$k > k_1 := 3 \lceil E \rceil \text{bins}(\lceil E \rceil), \quad \text{where } E = \max \left\{ 9, \frac{2d}{a_{1,t}} \cdot \frac{s-t}{\lambda_1 - \lambda_2} \right\}$$

because in this case $\left(\frac{2d}{a_{1,t}} \right)^{\frac{1}{s-t}} \leq \frac{2d}{a_{1,t}}$ and $\left(\frac{2d}{a_{1,t}} \right)^{-\frac{1}{s-t}} \leq 1$. On the other hand, if $2d < a_{1,t}$, then (9) holds when

$$k > k_1 := 3 \left\lceil \frac{a_{1,t}}{2d} E \right\rceil \text{bins}(\lceil E \rceil), \quad \text{where } E = \max \left\{ 9, \frac{s-t}{\lambda_1 - \lambda_2} \right\}$$

because in this case $\left(\frac{2d}{a_{1,t}} \right)^{-\frac{1}{s-t}} < \left(\frac{2d}{a_{1,t}} \right)^{-1}$ and $\left(\frac{2d}{a_{1,t}} \right)^{\frac{1}{s-t}} < 1$.

Finally, we conclude that (7) holds for all $k > k_2 := \max\{k_0, k_1\}$, where both k_0 and k_1 are computable in PTIME. In other words, we obtained a polynomially computable value k_2 such that $\mathcal{P}(a^k) > c = \lambda$ for all $k > k_2$. Using the same argument as at the end of the proof of Lemma 8, we can show that the (strict) emptiness and λ -reachability problems are decidable in EXPTIME. \blacktriangleleft

We are now ready to give a proof of Theorem 4.

Proof of Theorem 4. Let $\mathcal{P} = (u, A, v)$ be a polynomially ambiguous unary PFA. By Lemma 5, we can compute in EXPTIME a set of d polynomially ambiguous unary PFAs $\{\mathcal{P}_s = (u_s, U, v') \mid 0 \leq s \leq d-1\}$ such that U is rational upper-triangular and

$$\mathcal{P}(a^{rd+s}) = \mathcal{P}_s(a^r) = u_s^T U^r v',$$

where $0 \leq s \leq d-1$.

Suppose λ is a given cutpoint. If we want to decide whether there exists k such that $\mathcal{P}(a^k) = \lambda$ (or $\mathcal{P}(a^k) \geq \lambda$), we can check for every s from 0 to $d-1$ whether there exists r such that $\mathcal{P}_s(a^r) = \lambda$ (or $\mathcal{P}_s(a^r) \geq \lambda$, respectively), which can be done in EXPTIME using Lemmas 8 and 9. Namely, we will use Lemma 8 if $\lambda \neq c_s$ and Lemma 9 if $\lambda = c_s$ for the current values of $s \in [0, d-1]$. Finally, we note that even though the value of d can be exponential in the input size, the whole procedure can still be done in EXPTIME. ◀

Skolem's problem is at least NP-hard [6] implying that the λ -reachability and emptiness problems are also NP-hard, at least for PFA of exponential ambiguity. Our next result shows that NP-hardness can be established even for unary PFAs of *finite ambiguity*.

► **Theorem 10.** *The λ -reachability and emptiness problems for unary finitely ambiguous Probabilistic Finite Automata $\mathcal{P} = (u, A, v)$ with $\{0, 1\}$ -matrix A are NP-hard.*

Proof. The NP-hardness of Skolem's problem was established in [6]. Specifically, Corollary 1.3 of [6] states that the problem of determining, for a given matrix $A \in \{0, 1\}^{n \times n}$ and row vectors $b, c \in \{0, 1\}^n$, if $b^T A^k c = 0$ for some $k \geq 0$ is NP-hard. Examination of the proof of this corollary shows that in fact \mathcal{P} is finitely ambiguous as we shall show.

The proof of Theorem 1.1 of [6] shows a reduction of 3SAT on m clauses with n letters to a unary rational expression E of the form:

$$E = \bigcup_{j=0}^k a^{z_j} (a^{r_j})^*,$$

where $k = \mathcal{O}(n^3 m)$ and $z_j, r_j = \mathcal{O}(n^6)$ as is not difficult to see from the proof in [6]. Notice then that each z_j, r_j represented in unary has a polynomial size in terms of the 3SAT instance and thus E also has a polynomial representation size.

We may then invoke Kleene's theorem [15] to state that the language recognised by E is also recognised by an NFA $\mathcal{P} = (b, \{A\}, c)$ which thus allows the derivation of Corollary 1.3 of [6]. Note that E is simply the union of rational expressions of the form $E_j = a^{z_j} (a^{r_j})^*$. Each E_j can be transformed to an NFA \mathcal{N}_j with $z_j + r_j + 1$ states $S_j = n_{0,j}, \dots, n_{z_j,j}, n_{z_j+1,j}, \dots, n_{z_j+r_j,j}$ with initial state $n_{0,j}$, final state $n_{z_j+1,j}$ and transition function $\delta : S_j \times \{a\} \rightarrow S_j$ given by $\delta(n_{i,j}, a) = n_{i+1,j}$ for $0 \leq i \leq z_j + r_j - 1$ and $\delta(n_{z_j+r_j,j}, a) = n_{z_j+1,j}$.

We may then form an NFA \mathcal{N} by $\mathcal{N} = \bigcup_{j=0}^k \mathcal{N}_j$ with the usual construction. In this case, \mathcal{N} has set of initial states $\{n_{0,j} \mid 1 \leq j \leq k\}$, set of final states $\{n_{z_j+1,j} \mid 1 \leq j \leq k\}$ and states in disjoint subsets S_j and $S_{j'}$ with $j \neq j'$ are not connected. This implies by the IDA property of [23] that \mathcal{N} is finitely ambiguous since there does not exist any state with two outgoing transitions (by which reasoning we also know that each row of \mathcal{N} 's transition matrix has exactly one entry 1 with all others 0). In fact one may see that \mathcal{N} is k -ambiguous with $k = \mathcal{O}(n^3 m)$. The number of states of \mathcal{N} is $d = \sum_{j=0}^k z_j + r_j + 1 = \mathcal{O}(n^9 m)$ which is polynomial in the 3SAT instance representation size.

We note that actually \mathcal{N} is already close to a PFA. Since each row is zero except for exactly one entry 1, matrix A is stochastic. We thus consider Probabilistic Finite Automaton $\mathcal{P} = (u, \{A\}, c)$ where $u = \frac{b}{|b|}$ is the initial (stochastic) vector. \mathcal{P} has polynomial ambiguity

15:12 Decision Questions for Probabilistic Automata on Small Alphabets

since \mathcal{N} does. Therefore, deciding if there exists $k \geq 0$ such that $\mathcal{P}(a^k) = 0$ or $\mathcal{P}(a^k) \leq 0$ is NP-hard to determine, proving NP-hardness of the λ -reachability and emptiness problems. Since we did not modify \mathcal{N} to derive \mathcal{P} other than to scale the initial vector, the degree of ambiguity is retained. \blacktriangleleft

► **Corollary 11.** *The λ -reachability and emptiness problems for unary polynomially ambiguous PFA $\mathcal{P} = (u, A, v)$ with $\{0, 1\}$ -matrix A are NP-complete.*

Proof. NP-hardness follows from Theorem 10 since finite ambiguity is a stronger property than polynomially ambiguity. To prove the NP upper bound, we will show that the algorithm in the proof of Theorem 4 can be done in NP. We again use Lemmas 5, 6, 8 and 9. Note that the value d from Lemma 5 can be exponential. However, its binary presentation has polynomial size. So, instead of cycling through all s from 0 to $d-1$, we can nondeterministically guess in polynomial time a value $s \in [0, d-1]$.

Next, we note that the values of k_0 in Lemma 8 and k_2 in Lemma 9 also have binary representations of polynomial size. Again, instead of checking every k in $[0, k_0]$ or $[0, k_2]$, we can nondeterministically guess k in polynomial time.

Finally, in the verification step of our algorithm we need to compute the matrices A^d , A^s and $(A^d)^k$. This can be done in polynomial time using exponentiation by squaring. Indeed, the exponentiation by squaring requires polynomially many steps. Also, any power of a stochastic $\{0, 1\}$ -matrix is also a stochastic $\{0, 1\}$ -matrix, so the entries of the power matrices do not grow in size. \blacktriangleleft

4 Binary PFA

The following theorem shows that the λ -reachability and emptiness problems are NP-hard for binary PFA of *polynomial ambiguity* with *commuting* transition matrices (and the matrices can be assumed fixed in the case of λ -reachability and nonstrict emptiness). The emptiness problem for non-commutative binary PFA over 25 states is known to be undecidable, at least over exponentially ambiguous PFA [12]. Emptiness is also undecidable for exponentially ambiguous commutative PFA, although with many more states and a larger alphabet [2].

► **Theorem 12.** *The λ -reachability and emptiness problems are NP-hard for binary probabilistic finite automata of polynomial ambiguity with commuting matrices of dimension 9 for λ -reachability, 37 for nonstrict emptiness, and 40 for strict emptiness. Moreover, the matrices can be assumed fixed for the λ -reachability and nonstrict emptiness problems.*

Proof. We use a reduction from the solvability of binary quadratic Diophantine equations. Namely, given an equation of the form $ax^2 + by - c = 0$, where $a, b, c \in \mathbb{N}$, it is NP-hard to determine if there exists $x, y \in \mathbb{N}$ satisfying the equation [16]. We begin with the λ -reachability problem before considering the emptiness problem.

λ -Reachability reduction. Let $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and note that $A^k = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$ and that $(A \otimes A)^k_{1,4} = (A^k \otimes A^k)_{1,4} = k^2$. We form a weighted automaton² W_1 on binary alphabet $\Sigma = \{h, g\}$ in the following way to encode $ax^2 + by$ (we will deal with c later). Let $W_1 = (u_1, \phi, v_1)$ where $u_1, v_1 \in \mathbb{N}^7$ and $\phi : \Sigma^* \rightarrow \mathbb{N}^{7 \times 7}$. We define $u_1 = (a, 0, 0, 0, b, 0, 0)^T$, $v_1 = (0, 0, 0, 1, 0, 1, 0)^T$ and $\phi(\ell) = \frac{1}{4}\phi'(\ell)$ for $\ell \in \{h, g\}$ with

² For our purposes here, by a *weighted automaton* we simply mean an automaton whose initial vector, final vector, and transition matrices are over nonnegative integers.

$$\phi'(h) = \left(\frac{(A \otimes A) \oplus I_2}{\mathbf{0}^6} \middle| \frac{t_1}{4} \right), \quad \phi'(g) = \left(\frac{I_4 \oplus A}{\mathbf{0}^6} \middle| \frac{t_2}{4} \right),$$

with $\mathbf{0}^k = (0, 0, \dots, 0) \in \mathbb{N}^k$, $t_1 = (0, 2, 2, 3, 3, 3)^T$ and $t_2 = (3, 3, 3, 3, 2, 3)^T$. We see then that each row of $\phi'(\ell)$ is nonnegative and sums to 4, thus $\phi(\ell)$ is stochastic for $\ell \in \{g, h\}$. Furthermore, by the mixed product property of the Kronecker product, we see that $((A \otimes A) \oplus I_2)^x = (A^x \otimes A^x) \oplus I_2$ and $(I_4 \oplus A)^y = I_4 \oplus A^y$ for $x, y \in \mathbb{N}$ and thus by the block upper triangular structure of $\phi'(h), \phi'(g)$, we see that

$$\phi'(h^x g^y) = \left(\frac{(A^x \otimes A^x) \oplus A^y}{\mathbf{0}^6} \middle| \frac{t_{xy}}{4^{x+y}} \right),$$

where t_{xy} is a nonnegative vector maintaining the row sum at 4^{x+y} . We now see that

$$u_1^T \phi(h^x g^y) v_1 = \frac{ax^2 + by}{4^{x+y}} \quad (11)$$

We define a second weighted automaton $W_2 = (u_2, \psi, v_2)$ with $u_2 = (c, 0)^T$, $v_2 = (0, 1)^T$ and $\psi : \Sigma^* \rightarrow \mathbb{N}^{2 \times 2}$ with $\psi(\ell) = \frac{1}{4} \psi'(\ell)$ for $\ell \in \{h, g\}$ defined thus: $\psi'(h) = \psi'(g) = \begin{pmatrix} 1 & 3 \\ 0 & 4 \end{pmatrix}$.

We therefore see that

$$u_2^T \psi(h^x g^y) v_2 = \frac{c(4^{x+y} - 1)}{4^{x+y}} = c \left(1 - \frac{1}{4^{x+y}}\right) \quad (12)$$

We now join W_1 and W_2 into a 9-state PFA $\mathcal{P} = (u, \gamma, v)$ where $u = \frac{1}{a+b+c}[u_1 | u_2]$, $v = [v_1 | v_2]$ and $\gamma(\ell) = \phi(\ell) \oplus \psi(\ell)$. Combining Eqns (11) and (12) we see that

$$\begin{aligned} u^T \gamma(h^x g^y) v &= \frac{1}{a+b+c} \left(\frac{ax^2 + by}{4^{x+y}} + c \left(1 - \frac{1}{4^{x+y}}\right) \right) \\ &= \frac{1}{a+b+c} \left(c + \frac{ax^2 + by - c}{4^{x+y}} \right) \end{aligned} \quad (13)$$

which equals $\frac{c}{a+b+c}$ if and only if $ax^2 + by - c = 0$. Note that $\gamma(h)$ and $\gamma(g)$ commute by their structure since clearly $(A \otimes A) \oplus I$ and $I_4 \oplus A$ commute, giving $(A \otimes A) \oplus A$ in both cases (as a consequence of the mixed product properties of Lemma 1) and the rightmost vector of the matrix simply retains the row sum at 1 for such a product since the matrices are stochastic. Both $\gamma(h)$ and $\gamma(g)$ are upper-triangular thus \mathcal{P} is polynomially ambiguous.

Nonstrict Emptiness reduction. We now show the proof of the emptiness problem. We showed that the λ -reachability problem is NP-hard by deriving a PFA \mathcal{P} over the binary alphabet $\{h, g\}$ such that $\mathcal{P}(h^x g^y)$ is given by Eqn. 13. We note however that a non solution to $ax^2 + by - c = 0$ can be positive or negative and thus we may be above or below the threshold $\frac{c}{a+b+c}$. This encoding thus cannot be used to show the NP-hardness of the *emptiness* problem.

Instead, we can use a similar encoding of the quartic polynomial given by $(ax^2 + by - c)^2 = a^2x^4 + 2abx^2y + b^2y^2 + c^2 - 2acx^2 - 2bcy$ with $a, b, c \in \mathbb{N}$. Note that we arranged the four positive terms first, followed by the two negative terms. Clearly $(ax^2 + by - c)^2$ is nonnegative and equals zero if and only if $ax^2 + by - c = 0$. We will derive a PFA \mathcal{P}_2 such that

$$\mathcal{P}_2(h^x g^y) = \frac{1}{z} \left((2ac + 2bc) + \frac{1}{16^{x+y}} (ax^2 + by + c)^2 \right),$$

15:14 Decision Questions for Probabilistic Automata on Small Alphabets

where $z = a^2 + 2ab + b^2 + c^2 + 2ac + 2bc$, with the property that $\mathcal{P}_2(h^x g^y) \geq \frac{2ac+2bc}{z}$ with equality if and only if $(ax^2 + by - c)^2 = 0$ which is NP-hard to determine. To this end, we compute the following four matrices $\{H_+, G_+, H_-, G_-\}$, the idea being that H_+ and G_+ will be used to compute the positive four terms and H_- and G_- will compute the negative terms:

$$\begin{aligned} H_+ &= \underbrace{(A \otimes A \otimes A \otimes A)}_{x^4} \oplus \underbrace{(A \otimes A \otimes I_2)}_{x^2 y} \oplus \underbrace{(I_2 \otimes I_2)}_{y^2} \oplus \underbrace{1}_1 \\ G_+ &= \underbrace{(I_2 \otimes I_2 \otimes I_2 \otimes I_2)}_{x^4} \oplus \underbrace{(I_2 \otimes I_2 \otimes A)}_{x^2 y} \oplus \underbrace{(A \otimes A)}_{y^2} \oplus \underbrace{1}_1 \\ H_- &= \underbrace{(A \otimes A)}_{x^2} \oplus \underbrace{I_2}_y \\ G_- &= \underbrace{(I_2 \otimes I_2)}_{x^2} \oplus \underbrace{A}_y \end{aligned}$$

and by the mixed product property of Kronecker products of Lemma 1),

$$\begin{aligned} H_+^x G_+^y &= (A^x \otimes A^x \otimes A^x \otimes A^x) \oplus (A^x \otimes A^x \otimes A^y) \oplus (A^y \otimes A^y) \oplus 1 \\ H_-^x G_-^y &= (A^x \otimes A^x) \oplus A^y \end{aligned}$$

Note that $H_+^x G_+^y$ and $H_-^x G_-^y$ each contain the positive and negative (respectively) term of $(ax^2 + by - c)^2$, excluding the coefficients, e.g. $(H_+^x G_+^y)_{1,16} = x^4$ and $(H_+^x G_+^y)_{17,24} = x^2 y$ etc. Note also that $H_+ G_+ = G_+ H_+$ and $H_- G_- = G_- H_-$ which also follows from the mixed product properties and thus matrices $\{H_+, G_+\}$ and $\{H_-, G_-\}$ commute.

As before, we may now increase the dimension of each matrix $\{H_+, H_-, G_+, G_-\}$ by 1 to ensure a common row sum (of 16 in this case) by adding a new column on the right hand side of each matrix, and then divide each matrix by this common value to give $\{H'_+, H'_-, G'_+, G'_-\}$ so that each of these matrices is row stochastic. Matrices $\{H'_+, G'_+\}$ and $\{H'_-, G'_-\}$ still commute since this change only has an effect on the final column of the matrix.

We now show how to handle each term of $(ax^2 + by - c)^2$. We first handle the positive terms. We define $u_1 = (a^2, 0, \dots, 0)^T \in \mathbb{Q}^{16}$, $u_2 = (2ab, 0, \dots, 0)^T \in \mathbb{Q}^8$, $u_3 = (b^2, 0, 0, 0)^T \in \mathbb{Q}^4$ and $u_4 = c^2$ and then let $u_+ = [u_1 | u_2 | u_3 | u_4 | 0] \in \mathbb{Q}^{30}$. We let $v_1 = (0, \dots, 0, 1)^T \in \mathbb{Q}^{16}$, $v_2 = (0, \dots, 0, 1)^T \in \mathbb{Q}^8$, $v_3 = (0, 0, 0, 1)^T \in \mathbb{Q}^4$ and $v_4 = 1$, and let $v_+ = [v_1 | v_2 | v_3 | v_4 | 0] \in \mathbb{Q}^{30}$. We then see that

$$\begin{aligned} & u_+^T (H'_+)^x (G'_+)^y v_+ \\ &= \frac{1}{16^{x+y}} (u_1^T (A^x \otimes A^x \otimes A^x \otimes A^x) v_1 + u_2^T (A^x \otimes A^x \otimes A^y) v_2 + u_3^T (A^y \otimes A^y) v_3 + u_4^T v_4) \\ &= \frac{1}{16^{x+y}} (a^2 x^4 + 2abx^2 y + b^2 y^2 + c^2) \end{aligned} \quad (14)$$

We next handle the negative terms, which is essentially accomplished by switching final and non-final states in the final state vectors to follow. Define $u_5 = (2ac, 0, 0, 0)^T \in \mathbb{Q}^4$ and $u_6 = (2bc, 0)^T \in \mathbb{Q}^2$ and let $u_- = [u_5 | u_6 | 0] \in \mathbb{Q}^7$. We let $v_5 = (0, 0, 0, 1)^T \in \mathbb{Q}^4$ and $v_6 = (0, 1)^T \in \mathbb{Q}^2$. Define $v_- = [v_5 | v_6 | 0] \in \mathbb{Q}^7$. We then see that

$$\begin{aligned} & u_-^T (H'_-)^x (G'_-)^y (\mathbf{1} - v_-) \\ &= (2ac + 2bc) - \frac{1}{16^{x+y}} (u_5^T (A^x \otimes A^x) v_5 + u_6^T A^y v_6 + 0) \\ &= (2ac + 2bc) - \frac{1}{16^{x+y}} (2acx^2 + 2bcy), \end{aligned} \quad (15)$$

where $\mathbf{1} = (1, 1, \dots, 1)^T \in \mathbb{Q}^7$. We used here the fact that $X\mathbf{1} = \mathbf{1}$ for a row stochastic matrix X . We finally define that $H = H'_+ \oplus H'_- \in \mathbb{Q}^{37 \times 37}$ and $G = G'_+ \oplus G'_- \in \mathbb{Q}^{37 \times 37}$, both of which are row stochastic and commute, and let $u_\star = \frac{[u_+ | u_-]}{z} \in \mathbb{Q}^{37}$ and $v_\star = [v_+ | (\mathbf{1} - v_-)] \in \mathbb{Q}^{37}$,

with $z = a^2 + 2ab + b^2 + c^2 + 2ac + 2bc$ to normalise vector u_* . We see then that u_* is a stochastic vector as required. We define the PFA $\mathcal{P}_2 = (u_*, \{H, G\}, v_*)$ and we can now compute that

$$\begin{aligned}
\mathcal{P}_2(h^x g^y) &= u_*^T H^x G^y v_* \\
&= u_*^T (H'_+ G'^y \oplus H'_- G'^y) v_* \\
&= \frac{1}{16^{x+y}} \left(\frac{[u_+ | u_-]^T}{z} \left(\begin{array}{c|c} \frac{H'_+ G'^y}{\mathbf{0}} & \begin{array}{c} * \\ 16^{x+y} \end{array} \\ \hline \mathbf{0} & \frac{H'_- G'^y}{\mathbf{0}} \begin{array}{c} * \\ 16^{x+y} \end{array} \end{array} \right) [v_+ | (\mathbf{1} - v_-)] \right) \\
&= \frac{1}{z 16^{x+y}} (u_+^T H'_+ G'^y v_+ + u_-^T H'_- G'^y (\mathbf{1} - v_-)) \\
&= \frac{1}{z} (u_+^T (H'_+)^x (G'_+)^y v_+ + u_-^T (H'_-)^x (G'_-)^y (\mathbf{1} - v_-)) \\
&= \frac{1}{z} \left((2ac + 2bc) + \frac{1}{16^{x+y}} (a^2 x^4 + 2abx^2 y + b^2 y^2 + c^2) - \frac{1}{16^{x+y}} (2acx^2 + 2bcy) \right) \\
&= \frac{1}{z} \left((2ac + 2bc) + \frac{1}{16^{x+y}} (ax^2 + by - c)^2 \right) \tag{16}
\end{aligned}$$

where $*$ denote the column vectors used to ensure row sums of 16^{x+y} and $\mathbf{0}$ denotes zero matrices of appropriate sizes. We also used Eqns (14) and (15).

Since $(ax^2 + by - c)^2$ is nonnegative, we see that $u_*^T H^x G^y v_* \geq \frac{2ac+2bc}{z}$ with equality if and only if $(ax^2 + by - c)^2 = 0$, which is NP-hard to determine. Therefore using cutpoint $\lambda = \frac{2ac+2bc}{z} \in \mathbb{Q} \cap [0, 1]$ means the (nonstrict) emptiness problem is NP-hard (i.e. does there exist $x, y \in \mathbb{N}$ such that $u_*^T H^x G^y v_* \leq \lambda$ is NP-hard). As before, matrices H and G are upper-triangular and commute by their structure, and therefore the result holds.

Strict Emptiness reduction. Finally we show how to handle the strict emptiness problem. We proceed with a technique inspired by [10]. By (16), if $\mathcal{P}_2(h^x g^y) = u_*^T H^x G^y v_* \neq \frac{1}{z}(2ac + 2bc)$, then $u_*^T H^x G^y v_* \geq \frac{1}{z} \left((2ac + 2bc) + \frac{1}{16^{x+y}} \right)$ therefore $\mathcal{P}_2(h^x g^y) \leq \frac{1}{z}(2ac + 2bc)$ if and only if $\mathcal{P}_2(h^x g^y) < \frac{1}{z} \left((2ac + 2bc) + \frac{1}{16^{x+y}} \right)$.

Let us adapt \mathcal{P}_2 in the following way to create a new PFA \mathcal{P}_3 . Note that \mathcal{P}_2 has 6 initial states (by u_*). We add three new states to \mathcal{P}_3 , denoted q_0, q_F and q_* . State q_0 is a new initial state of \mathcal{P}_3 which, for any input letter, has probability $\frac{1}{2 \cdot 6}$ of moving to each of the 6 initial states of \mathcal{P}_2 and probability $\frac{1}{2}$ to move to new state q_F . State q_F is a new final state that remains in q_F for any input letter with probability $1 - \frac{1}{16z}$ and moves to a new non-accepting absorbing sink state q_* with probability $\frac{1}{16z}$. We now see that for any $a \in \{h, g\}$:

$$\mathcal{P}_3(aw) = \frac{1}{2} \mathcal{P}_2(w) + \frac{1}{2} \left(1 - \frac{1}{16^{|w|} z^{|w|}} \right)$$

If there exists $w_1 = h^x g^y$ with $x, y \geq 0$ such that $\mathcal{P}_2(w_1) \leq \frac{1}{z}(2ac + 2bc)$ then $\mathcal{P}_2(w_1) = \frac{1}{z}(2ac + 2bc)$ and thus:

$$\mathcal{P}_3(aw_1) = \frac{1}{2} \left(\frac{1}{z}(2ac + 2bc) \right) + \frac{1}{2} \left(1 - \frac{1}{16^{|w_1|} z^{|w_1|}} \right) < \frac{1}{2} \left(\frac{1}{z}(2ac + 2bc) + 1 \right).$$

For any $w_2 = h^x g^y$ with $x, y \geq 0$ such that $\mathcal{P}_2(w_2) > \frac{1}{z}(2ac + 2bc)$ then $\mathcal{P}_2(w_2) \geq \frac{1}{z}(2ac + 2bc) + \frac{1}{16^{x+y}}$ by (16). Thus:

$$\mathcal{P}_3(aw_2) \geq \frac{1}{2} \left(\frac{1}{z}(2ac + 2bc) + \frac{1}{16^{|w_2|}} \right) + \frac{1}{2} \left(1 - \frac{1}{16^{|w_2|} z^{|w_2|}} \right) > \frac{1}{2} \left(\frac{1}{z}(2ac + 2bc) + 1 \right).$$

Thus determining if there exists $w = h^x g^y$ such that $\mathcal{P}_3(w) < \frac{1}{2} \left(\frac{1}{z}(2ac + 2bc) + 1 \right)$, i.e. the strict emptiness problem for \mathcal{P}_3 on cutpoint $\frac{1}{2} \left(\frac{1}{z}(2ac + 2bc) + 1 \right)$, is NP-hard. The modifications to \mathcal{P}_2 retain polynomial ambiguity since q_0 and q_F have no incoming (non self looping) edges and q_* has no outgoing edges, therefore property EDA does not hold. Commutativity of the PFA is unaffected since \mathcal{P}_3 is identical to \mathcal{P}_2 except for adding three new states, behaving identically for both input letters. Note that \mathcal{P}_3 has $37 + 3 = 40$ states. ◀

References

- 1 S. Akshay, Timos Antonopoulos, Joël Ouaknine, and James Worrell. Reachability problems for Markov chains. *Information Processing Letters*, 115(2):155–158, 2015.
- 2 Paul C. Bell. Polynomially ambiguous probabilistic automata on restricted languages. In *International Colloquium on Automata, Languages, and Programming*, number 105 in ICALP'19, pages 1–14, 2019.
- 3 Paul C. Bell and P. Semukhin. Decidability of cutpoint isolation for probabilistic finite automata on letter-bounded inputs. In *International Conference on Concurrency Theory*, number 22 in CONCUR'20, pages 1–16, 2020.
- 4 A. Bertoni, G. Mauri, and M. Torelli. Some recursively unsolvable problems relating to isolated cutpoints in probabilistic automata. In *Automata, Languages and Programming*, volume 52, pages 87–94, 1977.
- 5 V. Blondel and V. Canterini. Undecidable problems for probabilistic automata of fixed dimension. *Theory of Computing Systems*, 36:231–245, 2003.
- 6 V. D. Blondel and N. Portier. The presence of a zero in an integer linear recurrent sequence is NP-hard to decide. *Linear Algebra and its Applications*, pages 91–98, 2002.
- 7 L. Daviaud, M. Jurdzinski, R. Lazic, F. Mazowiecki, G. A. Pérez, and J. Worrell. When is containment decidable for probabilistic automata? In *International Colloquium on Automata, Languages, and Programming*, number 121 in ICALP'18, pages 1–14, 2018.
- 8 N. Fijalkow, C. Riveros, and J. Worrell. Probabilistic automata of bounded ambiguity. In *28th International Conference on Concurrency Theory (CONCUR)*, pages 19:1–19:14, 2017.
- 9 S. Friedland. *Matrices: Algebra, Analysis and Applications*. World Scientific Publishing Company Pte Limited, 2015. URL: <https://books.google.co.uk/books?id=y8fACwAAQBAJ>.
- 10 H. Gimbert and Y. Oualhadj. Probabilistic automata on finite words: decidable and undecidable problems. In *International Colloquium on Automata, Languages and Programming (ICALP'10)*, volume 2, pages 527–538, 2010.
- 11 V. Halava, T. Harju, M. Hirvensalo, and J. Karhumäki. Skolem's problem — on the border between decidability and undecidability. In *TUCS Technical Report Number 683*, 2005.
- 12 M. Hirvensalo. Improved undecidability results on the emptiness problem of probabilistic and quantum cut-point languages. *SOFSEM 2007: Theory and Practice of Computer Science, Lecture Notes in Computer Science*, 4362:309–319, 2007.
- 13 R. A. Horn and C. R. Johnson. *Topics in matrix analysis*. Cambridge University Press, 1991.
- 14 O. Ibarra and B. Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In *Proc. STACS 1986*, volume 210, pages 171–179, 1986.
- 15 S. C. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies, Annals of Mathematical Studies*, 34, 1956.
- 16 K. L. Manders and L. Adleman. NP-complete decision problems for binary quadratics. *Journal of Computer and System Sciences*, 16:168–184, 1978.
- 17 J. Ouaknine and J. Worrell. Positivity problems for low-order linear recurrence sequences. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'14*, pages 366–379, SODA, 2014.
- 18 A. Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.
- 19 M. O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

- 20 C. Reutenauer. *Propriétés arithmétiques et topologiques de séries rationnelles en variables non commutatives*. Thèse troisième cycle, Université Paris VI, 1977.
- 21 P. Turakainen. Generalized automata and stochastic languages. *Proceedings of the American Mathematical Society*, 21:303–309, 1969.
- 22 N. K. Vereshchagin. The problem of appearance of a zero in a linear recurrence sequence (in russian). *Mat. Zametki*, 38(2), 1985.
- 23 A. Weber and H. Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991.

Ideal Membership Problem for Boolean Minority and Dual Discriminator

Arpitha P. Bharathi ✉

IDSIA-USI, Lugano, Switzerland

Monaldo Mastrolilli ✉

IDSIA-SUPSI, Lugano, Switzerland

Abstract

The polynomial Ideal Membership Problem (IMP) tests if an input polynomial $f \in \mathbb{F}[x_1, \dots, x_n]$ with coefficients from a field \mathbb{F} belongs to a given ideal $I \subseteq \mathbb{F}[x_1, \dots, x_n]$. It is a well-known fundamental problem with many important applications, though notoriously intractable in the general case. In this paper we consider the IMP for polynomial ideals encoding combinatorial problems and where the input polynomial f has degree at most $d = O(1)$ (we call this problem IMP_d).

A dichotomy result between “hard” (NP-hard) and “easy” (polynomial time) IMPs was achieved for Constraint Satisfaction Problems over finite domains [6, 34] (this is equivalent to IMP_0) and IMP_d for the Boolean domain [23], both based on the classification of the IMP through functions called polymorphisms. For the latter result, there are only six polymorphisms to be studied in order to achieve a full dichotomy result for the IMP_d . The complexity of the IMP_d for five of these polymorphisms has been solved in [23] whereas for the ternary minority polymorphism it was incorrectly declared in [23] to have been resolved by a previous result. In this paper we provide the missing link by proving that the IMP_d for Boolean combinatorial ideals whose constraints are closed under the minority polymorphism can be solved in polynomial time. This completes the identification of the precise borderline of tractability for the IMP_d for constrained problems over the Boolean domain. We also prove that the proof of membership for the IMP_d for problems constrained by the dual discriminator polymorphism over any finite domain can also be found in polynomial time. Bulatov and Rafiey [8] recently proved that the IMP_d for this polymorphism is decidable in polynomial time, without needing a proof of membership. Our result gives a proof of membership and can be used in applications such as Nullstellensatz and Sum-of-Squares proofs.

2012 ACM Subject Classification Mathematics of computing → Gröbner bases and other special bases; Mathematics of computing → Combinatoric problems

Keywords and phrases Polynomial ideal membership, Polymorphisms, Gröbner basis theory, Constraint satisfaction problems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.16

Related Version *Full Version*: <https://arxiv.org/abs/2006.16422> [4]

Funding This research was supported by the Swiss National Science Foundation project 200020-169022 “Lift and Project Methods for Machine Scheduling Through Theory and Experiments”.

1 Introduction

The study of polynomial ideals and related algorithmic problems goes back to David Hilbert [17]. The methods developed in this area to date find a wide range of applications in mathematics and computer science. In this paper we consider the polynomial Ideal Membership Problem, where we want to decide if a given polynomial belongs to a given ideal. This problem is a fundamental algorithmic problem with important applications in solving polynomial systems (see e.g. [12]), polynomial identity testing [12, 30] and underlies proof systems such as Nullstellensatz and Polynomial Calculus (see e.g. [2, 9, 15]).



© Arpitha P. Bharathi and Monaldo Mastrolilli;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 16; pp. 16:1–16:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

16:2 Ideal Membership Problem for Boolean Minority and Dual Discriminator

To introduce the problem formally, let $\mathbb{F}[x_1, \dots, x_n]$ be the ring of polynomials over a field \mathbb{F} with indeterminates x_1, \dots, x_n . A polynomial *ideal* I is a subset of the polynomial ring $\mathbb{F}[x_1, \dots, x_n]$ with two properties: for any two polynomials f, g in I , $f + g$ also belongs to I and so does hf for any polynomial h . By the Hilbert Basis Theorem [16] every ideal I has a finite generating set $F = \{f_1, \dots, f_r\} \subset I$ such that for every $f \in \mathbb{F}[x_1, \dots, x_n]$, we have $f \in I$ if and only if there is an “ideal membership proof”, namely a set of polynomials $\{h_1, \dots, h_r\} \subset \mathbb{F}[x_1, \dots, x_n]$ such that $f = h_1 f_1 + \dots + h_r f_r$. The polynomial IDEAL MEMBERSHIP PROBLEM (IMP) is to find out if a polynomial f belongs to an ideal I or not, given a set F of generators of the ideal (we use IMP_d to denote IMP when the input polynomial f has degree at most $d = O(1)$). The IMP is, in general, notoriously intractable. The results of Mayr and Meyer show that it is EXPSPACE-complete [24, 25].

Semidefinite programming (SDP) relaxations have been a powerful technique for approximation algorithm design ever since Goemans and Williamson celebrated result of Max-Cut [14]. With the aim to construct stronger and stronger SDP relaxations, the Sum-of-Squares (SOS) hierarchy has emerged as the most promising set of relaxations (see e.g. [20]). However, we still do not know the answer to even very basic questions about its power. For example, we do not even know when SOS is guaranteed to run in polynomial time! As recently observed by O’Donnell [26], bounded degree SOS proof does not necessarily imply its low bit complexity, showing that the often repeated claim, that for any fixed degree SOS runs in polynomial time, is far from true. O’Donnell raised the open problem to establish useful conditions under which “small” SOS proof can be guaranteed. With this aim, a first elegant sufficient condition is due to Raghavendra and Weitz [27, 33]. For each instance \mathcal{C} of a combinatorial problem, the set of polynomials that vanish at every point of the set of solutions of \mathcal{C} is called the *combinatorial ideal* of \mathcal{C} and denoted by $I_{\mathcal{C}}$. To satisfy Raghavendra and Weitz’s criterion, it is necessary (but also sufficient) that the ideal membership problem IMP_d for each $I_{\mathcal{C}}$ is polynomial time solvable and that ideal membership proofs can be efficiently found too.¹ So the tractability of the ideal membership proof ensures that SOS runs in polynomial time for combinatorial problems. This is currently the only known general result that addresses the SOS bit complexity issue. However, the IMP_d tractability criterion of Raghavendra and Weitz suffers from a severe limitation, namely it is not clear which restrictions on combinatorial problems can guarantee an efficient computation of the IMP_d proofs for combinatorial ideals.

The Constraint Satisfaction Problem (CSP) provides a general framework for a wide range of combinatorial problems, where we are given a set of variables and a set of constraints, and we have to decide whether the variables can be assigned values that satisfy the constraints. There are useful connections between IMP_d and the CSP: for example a CSP instance \mathcal{C} is unsatisfiable if and only if $1 \in I_{\mathcal{C}}$. It follows that CSP is just the special case of IMP_d with $d = 0$ (see Appendix A.1 for more details on Ideal-CSP correspondence). Restrictions on CSPs, called $\text{CSP}(\Gamma)$, in which the type of constraints is limited to relations from a set Γ , have been successfully applied to study the computational complexity classification (and other algorithmic properties) of CSPs (see [7] for an excellent survey).

Motivated by the aforementioned issue of Raghavendra and Weitz criterion, Mastrolilli [23] initiated a systematic study of the IMP_d tractability for combinatorial ideals of the form $\text{IMP}_d(\Gamma)$ arising from combinatorial problems from $\text{CSP}(\Gamma)$ for a set of relations Γ over the Boolean domain. The classic dichotomy result of Schaefer [29] gives the complexity of $\text{CSP}(\Gamma)$ (and therefore of $\text{IMP}_0(\Gamma)$) for the Boolean domain: $\text{CSP}(\Gamma)$ is solvable in polynomial time

¹ Note that answering whether a polynomial belongs to a certain ideal does not necessarily mean finding an ideal membership proof of that.

if all constraints are closed under one of six polymorphisms (majority, minority, MIN, MAX, constant 0 and constant 1), else it is NP-complete. Mastrolilli [23] claimed a dichotomy result for the $\text{IMP}_d(\Gamma)$ for the Boolean domain: for any constant $d \geq 1$, the $\text{IMP}_d(\Gamma)$ of Boolean combinatorial ideals is solvable in polynomial time if all constraints are closed under one of four polymorphisms (majority, minority, MIN, MAX), else it is coNP-complete. In [23], for three polymorphisms (majority, MIN, MAX), it is shown that $\text{IMP}_d(\Gamma)$ is polynomial time solvable, and moreover ideal membership proofs can be efficiently found, too. Whereas for the ternary minority polymorphism it was incorrectly declared to have been resolved by a previous result². As a matter of fact the complexity of the $\text{IMP}_d(\Gamma)$ for the ternary minority polymorphism is open. It was mistakenly assumed in [23] that computing the (mod 2) Gröbner basis in lexicographic order was sufficient to solve the IMP_d problem in polynomial time, but the issue is that we require polynomials to be over \mathbb{R} and not $\text{GF}(2)$.

We address these issues in this paper and therefore establish the full dichotomy result claimed in [23] (see [22] for an updated version of the paper). To ensure efficiency of the $\text{IMP}_d(\Gamma)$ for the ternary minority polymorphism, it is sufficient to compute a d -truncated Gröbner basis in the graded lexicographic order (see Definition 8 and Appendix A for more details). This is achieved by first showing that we can easily find a Gröbner basis in the lexicographic order for the combinatorial ideal. Since polynomials in this Gröbner basis can have degrees up to n and coefficients of exponential size, we show how this basis can be converted to a d -truncated Gröbner basis in the graded lexicographic order in polynomial time, whose polynomials have degrees up to d and coefficients of constant size. This efficiently solves the $\text{IMP}_d(\Gamma)$ for combinatorial ideals whose constraints are over a language Γ closed under the minority polymorphism. Together with the results in [23, 22], our result allows to complete the answer of the aforementioned question by allowing to identify the precise borderline of tractability of the Boolean $\text{IMP}_d(\Gamma)$. Thus the following summarizes our first result of this paper:

► **Theorem 1.** *Let Γ be a constraint language over the Boolean domain that is closed under the minority polymorphism. For each instance \mathcal{C} of $\text{CSP}(\Gamma)$, the d -truncated reduced Gröbner basis in the graded lexicographic monomial ordering of the combinatorial ideal $I_{\mathcal{C}}$ can be computed in $n^{O(d)}$ time.*

► **Corollary 2.** *If Γ is closed under the minority polymorphism, then the ideal membership proofs of $\text{IMP}_d(\Gamma)$ over the Boolean domain can be computed in polynomial time for $d = O(1)$.*

After the appearance of a preliminary version of this paper [4], Bulatov and Rafiey [8] have recently obtained exciting new results. For a finite domain $D = \{0, 1, \dots, p-1\}$ with prime p elements, they consider the affine polymorphism $\otimes : D^3 \rightarrow D$ defined as $\otimes(a, b, c) = a - b + c \pmod{p}$ (it is easy to see that this is the minority polymorphism for the Boolean domain). By building on our approach, they prove that a d -truncated Gröbner basis can be computed in time $n^{O(d)}$ for any fixed prime p .

In [3], we began the generalization of $\text{CSP}(\Gamma)$ (viz. $\text{IMP}_0(\Gamma)$) by working on the corresponding $\text{IMP}_d(\Gamma)$ for any $d = O(1)$ in the ternary domain, which expands the known set of tractable IMP_d cases by providing a suitable class of combinatorial problems. We considered problems constrained under the dual discriminator polymorphism and prove that we can find the reduced Gröbner basis of the corresponding combinatorial ideal in polynomial time. This ensures that we can check if any degree d polynomial belongs to the combinatorial ideal or not in polynomial time, and provide proof of membership if it does. Among the very interesting results obtained in [8], the authors show that the IMP_d is solvable in polynomial

² This was pointed out by Andrei Bulatov, Akbar Rafiey and Stanislav Živný.

time for *any* finite domain for problems constrained under the dual discriminator. This was done by eliminating permutation constraints in some sense through a pre-processing step and converting an instance $\mathcal{C} = (X, D, C)$ to an instance $\mathcal{C}' = (X', D, C')$ where $X' \subseteq X$ and $C' \subseteq C$. Moreover, a polynomial $f(X)$ was converted to a polynomial $f'(X')$ such that $f \in I_{\mathcal{C}}$ if and only if $f' \in I_{\mathcal{C}'}$. They calculated a Gröbner basis of $I_{\mathcal{C}'}$, in polynomial time, which reflected the remaining constraints. This gives a proof of membership of f' in $I_{\mathcal{C}'}$ if it does belong to the ideal, but it is not yet known as to how to recover the proof of membership for f in $I_{\mathcal{C}}$. Meanwhile our results in [3] gives proof of membership, but is only constrained to a 3-element domain.

In this paper, we compute a Gröbner basis for the entire combinatorial ideal over a finite domain by showing that a Gröbner basis of the ideal associated with permutation constraints can also be calculated in polynomial time. We forego the pre-processing step of [8], include the permutation constraints and directly calculate a Gröbner basis of $I_{\mathcal{C}}$. The set of polynomials that the elements of the Gröbner basis can come from is polynomial in size and hence we show that a proof of membership can also be calculated in polynomial time as required in [28]. The following summarizes the second result of this paper:

► **Theorem 3.** *Let Γ be a constraint language over a finite domain D that is closed under the dual discriminator polymorphism. For each instance \mathcal{C} of $\text{CSP}(\Gamma)$, a Gröbner basis in the graded lexicographic monomial ordering of the combinatorial ideal $I_{\mathcal{C}}$ can be computed in time polynomial in the number of variables. The polynomials in this basis have degree at most $|D|$.*

► **Corollary 4.** *If Γ is closed under the dual discriminator polymorphism, then membership proofs for $\text{IMP}(\Gamma)$, over a finite domain, can be computed in polynomial time.*

The study of CSP-related IMPs is in its early stages. The results obtained in this paper are steps towards the long term and challenging goal of extending the celebrated dichotomy results of $\text{CSP}(\Gamma)$ for finite domain [6, 34] to $\text{IMP}(\Gamma)$. This would provide a complete CSP-related characterization of when the IMP tractability criterion is applicable.

Due to space limitations, we provide a sketch for some of the proofs, and the complete proofs will be updated in [4]. To make the paper more self-contained, some essential background and standard (according to the book [12]) Gröbner basis notations can be found in Appendix A.

2 Preliminaries

Let D denote a finite set (*domain*). By a k -ary **relation** R on a domain D we mean a subset of the k -th cartesian power D^k ; k is said to be the *arity* of the relation. We often use relations and (affine) varieties interchangeably since both essentially represent a set of solutions. A **constraint language** Γ over D is a set of relations over D . A constraint language is **finite** if it contains finitely many relations, and is **Boolean** if it is over the 2-element domain $\{0, 1\}$. A **constraint** over a constraint language Γ is an expression of the form $R(x_1, \dots, x_k)$ where R is a relation of arity k contained in Γ , and the x_i are variables. A constraint is satisfied by a mapping ϕ defined on the x_i if $(\phi(x_1), \dots, \phi(x_k)) \in R$.

► **Definition 5.** *The (nonuniform) CONSTRAINT SATISFACTION PROBLEM (CSP) associated with language Γ over D is the problem $\text{CSP}(\Gamma)$ in which: an instance is a triple $\mathcal{C} = (X, D, C)$ where $X = \{x_1, \dots, x_n\}$ is a set of n variables and C is a set of constraints over Γ with variables from X . The goal is to decide whether or not there exists a solution, i.e. a mapping $\phi : X \rightarrow D$ satisfying all of the constraints. We will use $\text{Sol}(\mathcal{C})$ to denote the set of solutions of \mathcal{C} .*

Moreover, we follow the algebraic approach to Schaefer's dichotomy result [29] formulated by Jeavons [18] where each class of CSPs that are polynomial time solvable is associated with a polymorphism.

► **Definition 6.** An operation $f : D^m \rightarrow D$ is a **polymorphism** of a relation $R \subseteq D^k$ if for any choice of m tuples from R (allowing repetitions), it holds that the tuple obtained from these m tuples by applying f coordinate-wise is in R . If this is the case we also say that f preserves R , or that R is invariant or closed with respect to f . A polymorphism of a constraint language Γ is an operation that is a polymorphism of every $R \in \Gamma$.

For a given instance \mathcal{C} of $\text{CSP}(\Gamma)$, the vanishing ideal of its solution set, $\mathbf{I}(\text{Sol}(\mathcal{C}))$, is called its **combinatorial ideal** and is denoted by $I_{\mathcal{C}}$ (see Definition 20 in Appendix A). We call polynomials of the form $\prod_{a \in D} (x_i - a)$ **domain polynomials**, denoted by $\text{dom}(x_i)$. They describe the fact that $\text{Sol}(\mathcal{C}) \subseteq D^n$. For a more detailed Ideal-CSP correspondence we refer to Appendix A.1.

► **Definition 7.** The IDEAL MEMBERSHIP PROBLEM associated with language Γ is the problem $\text{IMP}(\Gamma)$ in which the input consists of a polynomial $f \in \mathbb{F}[X]$ and a $\text{CSP}(\Gamma)$ instance $\mathcal{C} = (X, D, C)$. The goal is to decide whether f lies in the combinatorial ideal $I_{\mathcal{C}}$. We use $\text{IMP}_d(\Gamma)$ to denote $\text{IMP}(\Gamma)$ when the input polynomial f has degree at most d .

The Gröbner basis G of an ideal is a set of generators such that $f \in \langle G \rangle \iff f|_G = 0$, where $f|_G$ denotes the remainder of f divided by G (see [12] or Appendix A.2 for more details and notations).

► **Definition 8.** If G is a Gröbner basis of an ideal in $\mathbb{F}[x_1, \dots, x_n]$, the **d -truncated Gröbner basis** G' of G is defined as

$$G' = G \cap \mathbb{F}[x_1, \dots, x_n]_d,$$

where $\mathbb{F}[x_1, \dots, x_n]_d$ is the set of polynomials of degree less than or equal to d .

It is not necessary to compute a Gröbner basis of $I_{\mathcal{C}}$ in its entirety to solve the IMP_d . Since the input polynomial f has degree $d = O(1)$, the only polynomials from G that can possibly divide f in the graded lexicographic order (see Definition 26 in Appendix A.2), are those that are in G' . The remainders of such divisions are also in $\mathbb{F}[x_1, \dots, x_n]_d$. Therefore, by Proposition 32 and Corollary 33, the membership test can be computed by using only polynomials from G' and therefore we have

$$f \in I_{\mathcal{C}} \cap \mathbb{F}[x_1, \dots, x_n]_d \iff f|_{G'} = 0.$$

From the previous observations it follows that if we can compute G' in $n^{O(d)}$ time then this yields an algorithm that runs in $n^{O(d)}$ time for the IMP_d (note that the size of the input polynomial f is bounded by $n^{O(d)}$).

3 Boolean Minority

The Boolean Minority polymorphism is an affine polymorphism defined as follows. Note that there is only one such polymorphism for the Boolean domain.

► **Definition 9.** For a finite domain D , a ternary operation \otimes is called a **minority polymorphism** if $\otimes(a, a, b) = \otimes(a, b, a) = \otimes(b, a, a) = b$ for all $a, b \in D$.

3.1 Gröbner bases in lex order

Consider an instance $\mathcal{C} = (X = \{x_1, \dots, x_n\}, D = \{0, 1\}, C)$ of $\text{CSP}(\Gamma)$ where Γ is \otimes -closed. Any constraint of \mathcal{C} can be written as a system of linear equations over $\text{GF}(2)$ (see e.g. [10]). These linear systems with variables x_1, \dots, x_n can be solved by Gaussian elimination. If there is no solution, then we have from Hilbert's Weak Nullstellensatz (Theorem 25) that $1 \in I_{\mathcal{C}} \iff \text{Sol}(\mathcal{C}) = \emptyset \iff I_{\mathcal{C}} = \mathbb{R}[\mathbf{x}]$. If $1 \in I_{\mathcal{C}}$ the reduced Gröbner basis is $\{1\}$. We proceed only if $\text{Sol}(\mathcal{C}) \neq \emptyset$. In this section, we assume the lex order $>_{\text{lex}}$ with $x_1 >_{\text{lex}} x_2 >_{\text{lex}} \dots >_{\text{lex}} x_n$. We also assume that the linear system has $r \leq n$ equations and is already in its reduced row echelon form with x_i as the leading monomial of the i -th equation. Let $\text{Supp}_i \subset [n]$ such that $\{x_j : j \in \text{Supp}_i\}$ is the set of variables appearing in the i -th equation of the linear system except for x_i . Let the i -th equation be $R_i = 0 \pmod{2}$ where

$$R_i := x_i \oplus f_i, \quad (1)$$

with $i \in [r]$ and f_i is the Boolean function $(\bigoplus_{j \in \text{Supp}_i} x_j) \oplus \alpha_i$ and $\alpha_i = 0/1$.

3.2 From (mod 2) to regular arithmetic Gröbner basis

In this section, we show how to transform R_i 's into polynomials in regular arithmetic. The idea is to map R_i to a polynomial R'_i over $\mathbb{R}[x_1, \dots, x_n]$ such that $a \in \{0, 1\}^n$ satisfies $R_i = 0$ if and only if a satisfies $R'_i = 0$. Moreover, R'_i is such that it has the same leading term as R_i . We produce a set of polynomials G_1 and prove that G_1 is the reduced Gröbner basis of $I_{\mathcal{C}}$ over $\mathbb{R}[x_1, \dots, x_n]$ in the lex ordering. We define R'_i as

$$R'_i := x_i - M(f_i) \quad (2)$$

where

$$M(f_i) = \begin{cases} \sum_{k=1}^{|\text{Supp}_i|} \left((-1)^{k-1} \cdot 2^{k-1} \sum_{\{x_{j_1}, \dots, x_{j_k}\} \subseteq \text{Supp}_i} x_{j_1} x_{j_2} \dots x_{j_k} \right) & \text{when } \alpha_i = 0 \\ 1 + \sum_{k=1}^{|\text{Supp}_i|} \left((-1)^k \cdot 2^{k-1} \sum_{\{x_{j_1}, \dots, x_{j_k}\} \subseteq \text{Supp}_i} x_{j_1} x_{j_2} \dots x_{j_k} \right) & \text{when } \alpha_i = 1 \end{cases} \quad (3)$$

► **Lemma 10.** *Consider the following set of polynomials:*

$$G_1 = \{R'_1, \dots, R'_r, x_{r+1}^2 - x_{r+1}, \dots, x_n^2 - x_n\}, \quad (4)$$

where R'_i is from Equation (2). G_1 is the reduced Gröbner basis of $I_{\mathcal{C}}$ in the lexicographic order $x_1 >_{\text{lex}} x_2 >_{\text{lex}} \dots >_{\text{lex}} x_n$.

Proof. For any two Boolean variables x and y ,

$$x \oplus y = x + y - 2xy. \quad (5)$$

By repeatedly using Equation (5) to obtain the equivalent expression for f_i , we see that $R_i = 0 \pmod{2}$ and $R'_i = 0$ have the same set of 0/1 solutions. Therefore $\mathbf{V}(\langle G_1 \rangle)$ is equal to $\text{Sol}(\mathcal{C})$. This implies that $\langle G_1 \rangle \subseteq I_{\mathcal{C}}$. Moreover, $\text{LM}(R_i) = \text{LM}(R'_i) = x_i$, by construction. For every pair of polynomials in G_1 the reduced S -polynomial is zero as the leading monomials of any two polynomials in G_1 are relatively prime. By Buchberger's Criterion (see Theorem 36) it follows that G_1 is a Gröbner basis of $\langle G_1 \rangle$ over $\mathbb{R}[x_1, \dots, x_n]$ (according to the lex order).

In fact, it can be seen by inspection that G_1 is the *reduced* Gröbner basis of $\langle G_1 \rangle$. To prove that $I_{\mathcal{C}} = \langle G_1 \rangle$, we need to prove that any $p \in I_{\mathcal{C}} \implies p \in \langle G_1 \rangle$. It is enough to prove that $p|_{G_1} = 0$ as this implies $p \in \langle G_1 \rangle$. We have that $p|_{G_1}$ cannot contain variable x_i for all $1 \leq i \leq r$. Hence $p|_{G_1}$ is multilinear in $x_{r+1}, x_{r+2}, \dots, x_n$. Each tuple of D^{n-r} extends to exactly that n -tuple in $Sol(\mathcal{C})$ whose coordinate associated with x_i ($1 \leq i \leq r$) is the unique value x_i takes to satisfy $x_i \oplus f_i = 0$ (see Equation (1) and Equation (2)). As $p|_{G_1}$ is multilinear in $x_{r+1}, x_{r+2}, \dots, x_n$, there are at most 2^{n-r} coefficients. Since every point of D^{n-r} is a solution of $p|_{G_1}$, we see that every coefficient of $p|_{G_1}$ is zero and hence $p|_{G_1}$ is the zero polynomial. Hence G_1 is the reduced Gröbner basis of $I_{\mathcal{C}}$. ◀

Note that the reduced Gröbner basis in Equation (4) can be “efficiently” computed by exploiting the high degree of symmetry in each $M(f_i)$ and using elementary symmetric polynomials with variables from $Supp_i$.

3.3 Computing a truncated Gröbner basis

Now that we have the reduced Gröbner basis in lex order, we show how to obtain the d -truncated reduced Gröbner basis in \mathbf{glex} order in polynomial time for any fixed $d = O(1)$. Before we describe our conversion algorithm, we show how to expand a product of Boolean functions. This expansion will play a crucial step in our algorithm.

3.3.1 Expansion of a product of Boolean functions

In this section, we show a relation between a product of Boolean functions and (mod 2) sums of the Boolean functions, which is heavily used in our conversion algorithm in Section 3.3.2. We have already seen from Equation (5) that if f, g are two Boolean functions,³ then

$$2 \cdot f \cdot g = f + g - (f \oplus g).$$

Hence it can be proved by repeated use of the above equation that the following holds for Boolean functions f_1, f_2, \dots, f_m :

$$f_1 \cdot f_2 \cdots f_m = \frac{1}{2^{m-1}} \left[\sum_{i \in [m]} f_i - \sum_{\{i,j\} \subset [m]} (f_i \oplus f_j) + \sum_{\{i,j,k\} \subset [m]} (f_i \oplus f_j \oplus f_k) + \cdots + (-1)^{m-1} (f_1 \oplus f_2 \oplus \cdots \oplus f_m) \right]. \quad (6)$$

We call each Boolean function of the form $(f_{i_1} \oplus \cdots \oplus f_{i_k})$ in Equation (6) as a **Boolean term**. We call the Boolean term $(f_1 \oplus f_2 \oplus \cdots \oplus f_m)$ as the **longest Boolean term** of the expansion. Thus, a product of Boolean functions can be expressed as a linear combination of Boolean terms. Note that Equation (6) is *symmetric* with respect to f_1, f_2, \dots, f_m as any f_i interchanged with f_j produces the same expression. It is no coincidence that we chose the letter f in the above equation: we later apply this identity using f_j from $R_j := x_j \oplus f_j$ (see Section 3.1). When we use Equation (6) in the conversion algorithm, we will have to evaluate a product of at most d functions, i.e. $m \leq d = O(1)$. We now see in the right hand side of Equation (6) that the coefficient $1/2^{m-1}$ is of constant size and there are $O(1)$ many Boolean terms.

³ We earlier considered Boolean variables, but the same holds for Boolean functions.

3.3.2 Our conversion algorithm

The FGLM [13] conversion algorithm is well known in computer algebra for converting a given reduced Gröbner basis of a zero dimensional ideal in some ordering to the reduced Gröbner basis in any other ordering. However, it does so with $O(nD(\langle G_1 \rangle)^3)$ many arithmetic operations, where $D(\langle G_1 \rangle)$ is the dimension of the \mathbb{R} -vector space $\mathbb{R}[x_1, \dots, x_n]/\langle G_1 \rangle$ (see Proposition 4.1 in [13]). $D(\langle G_1 \rangle)$ is also equal to the number of common zeros (with multiplicity) of the polynomials from $\langle G_1 \rangle$, which would imply that for the combinatorial ideals considered in this paper, $D(\langle G_1 \rangle) = O(2^{n-r})$. This exponential running time is avoided in our conversion algorithm, which is a variant the FGLM algorithm, by exploiting the symmetries in Equation (3) and by truncating the computation up to degree d .

Some notations necessary for the algorithm are as follows: G_1 and G_2 are the reduced Gröbner basis of $\langle G_1 \rangle$ in `lex` and `grlex` ordering respectively. $\text{LM}(G_i)$ is the set of leading monomials of polynomials in G_i for $i \in \{1, 2\}$. Since we know G_1 , we know $\text{LM}(G_1)$, whereas G_2 and $\text{LM}(G_2)$ are constructed by the algorithm. $B(G_1)$ is the set of monomials that cannot be divided (considering the `lex` order) by any monomial of $\text{LM}(G_1)$. Therefore, $B(G_1)$ is the set of all multilinear monomials in variables x_{r+1}, \dots, x_n . Similarly, $B(G_2)$ is the set of monomials that cannot be divided (considering the `grlex` order) by any monomial of $\text{LM}(G_2)$. Recall the definition of f_i for $i \leq r$ from Section 3.1. For $i > r$, for notational purposes, we define the Boolean function $f_i := x_i$.

► **Lemma 11.** *Consider a monomial q such that $\deg(q) \leq d$. Then $q|_{G_1}$ can be expressed as a linear combination of Boolean terms.*

Proof. Consider $q = x_{i_1}x_{i_2} \cdots x_{i_k}$ where $k \leq d$. Then from Equations (1) and (2), $q|_{G_1} = f_{i_1}f_{i_2} \cdots f_{i_k}$ and the lemma holds using Equation (6). ◀

Let elements b_i of $B(G_2)$ be arranged in increasing `grlex` order. We construct a set A in our algorithm such that its elements a_i are defined as $a_i = b_i|_{G_1}$ written as linear combinations of Boolean terms using Lemma 11. We say that a Boolean term f of a_i “appears in a_j ” for some $j < i$ if the longest Boolean term of a_j is $f \oplus \alpha$ where $\alpha = 0/1$.

Let Q be the set of all monomials m such that $1 <_{\text{grlex}} \deg(m) \leq_{\text{grlex}} d$. We recommend the reader to refer to the example in [4] for an intuitive working of the algorithm. The conversion is described in full in Algorithm 1 (we assume $1 \notin I_C$, else $G_1 = \{1\} = G_2$ and we are done).

► **Lemma 12.** *The set A is such that every a_i is a linear combination of existing $b_j|_{G_1}$ ’s ($j < i$) and the longest Boolean term of $b_i|_{G_1}$.*

Proof. By definition, element a_i is added to A when a monomial q is added to $B(G_2)$ where $b_i = q$ and $a_i = b_i|_{G_1}$ expressed in Boolean terms (see Algorithm 1). This means that q is not divisible by any monomial in $\text{LM}(G_2)$. We prove the lemma by induction on the degree of q . Note that $b_1 = 1$ and hence $a_1 = b_1|_{G_1} = 1$.

If $\deg(q) = 1$, then q is some x_i and $x_i|_{G_1}$ is one of $0, 1$ or f_i . If $x_i|_{G_1}$ is either 0 or 1 , then it appears in a_1 . We are now in the “else” condition of Algorithm 1, so q should be added to $\text{LM}(G_2)$ and not $B(G_2)$. Hence $x_i|_{G_1}$ can be neither 0 nor 1 and the lemma holds for $\deg(q) = 1$ as f_i is the longest Boolean term.

Let us assume the statement holds true for all monomials with degree less than m . Consider q such that $\deg(q) = m$ and $q = x_{i_1}x_{i_2} \cdots x_{i_m}$ where i_j ’s need not be distinct, and the lemma holds for every monomial $<_{\text{grlex}} q$. Then $q|_{G_1} = f_{i_1} \cdot f_{i_2} \cdots f_{i_m}$. Let $(f_{j_1} \oplus \cdots \oplus f_{j_k})$ be a Boolean term in the expansion of $q|_{G_1}$ (by using Equation (6)), that

■ **Algorithm 1** Computing the d -truncated reduced Gröbner basis.

Input: Degree d , G_1 , Q .
Output: d -Truncated versions of G_2 , $B(G_2)$.
Initialization: $G_2 = \emptyset$, $B(G_2) = \{b_1 = 1\}$, $A = \{a_1 = 1\}$.

- 1 **while** $Q \neq \emptyset$ **do**
- 2 Let q be the smallest (according to grlex order) monomial in Q .
- 3 Find $q|_{G_1}$, by which we simply replace any occurrence of x_i by the Boolean functions f_i .
- 4 Expand $q|_{G_1}$ by using Equation (6).
- 5 **if** the longest Boolean term of $q|_{G_1}$ does not appear in any $a \in A$ **then**
- 6 Write $q|_{G_1}$ as a linear combination of $b_i|_{G_1}$ and its longest Boolean term (see Lemma 12).
- 7 Add this polynomial to A and add q to $B(G_2)$.
- 8 **else**
- 9 Every Boolean term of $q|_{G_1}$ can be written as linear combinations of $b_j|_{G_1}$'s. Note that if the longest Boolean term f appears in a as $f \oplus 1$, then we use $f \oplus 1 = 1 - (f)$ (see Equation (5)). Thus we have $q|_{G_1} = \sum_j k_j b_j|_{G_1} \implies q - \sum_j k_j b_j \in \langle G_1 \rangle$.
- 10 Add the polynomial $q - \sum_j k_j b_j$ to G_2 and q to $\text{LM}(G_2)$.
- 11 Delete any monomial in Q that q can divide.
- 12 Delete q from Q .
- 13 G_2 is the d -truncated reduced Gröbner basis.

is not the longest Boolean term, so $\{j_1, \dots, j_k\} \subset \{i_1, \dots, i_m\}$ and $k < m$. Consider the monomial $x_{j_1} x_{j_2} \dots x_{j_k}$. We will now prove that $x_{j_1} x_{j_2} \dots x_{j_k}$ is in fact some $b_l \in B(G_2)$ and there exists $a_l \in A$ which is a linear combination of $b_i|_{G_1}$'s and $(f_{j_1} \oplus \dots \oplus f_{j_k})$. The monomial $x_{j_1} x_{j_2} \dots x_{j_k}$ either belongs to $\text{LM}(G_2)$ or $B(G_2)$. If $x_{j_1} x_{j_2} \dots x_{j_k} \in \text{LM}(G_2)$ then it divides q , a contradiction to our choice of q . Therefore, $x_{j_1} x_{j_2} \dots x_{j_k} = b_l \in B(G_2)$. Clearly $b_l <_{\text{grlex}} q$ and the induction hypothesis applies, so there exists $a_l \in A$ such that

$$b_l|_{G_1} = a_l = \sum_{i < l} c_i b_i|_{G_1} + c_0 (f_{j_1} \oplus \dots \oplus f_{j_k})$$

where c_i 's are constants. Then we simply use the above equation to substitute for the Boolean term $f_{j_1} \oplus \dots \oplus f_{j_k}$ in $q|_{G_1}$ as a linear combination of $b_i|_{G_1}$ where $i \leq l$. We can do this for every Boolean term of $q|_{G_1}$ except the longest one. Hence the lemma holds. ◀

► **Theorem 13.** *The conversion algorithm terminates for every input G_1 and correctly computes a d -truncated reduced Gröbner basis, with the grlex ordering, of the ideal $\langle G_1 \rangle$ in polynomial time.*

Proof. Algorithm 1 runs at most $|Q| = O(n^d)$ times. Evaluation of any $q|_{G_1}$ can be done in $O(n)$ steps (see Equation (6)), checking if previous a_i 's appear (and replacing every Boolean term appropriately if it does) takes at most $O(n^d)$ steps since there are at most $|Q|$ many elements in A . Hence the running time of the algorithm is $O(n^{2d})$.

Suppose the set of polynomials $\{g_1, g_2, \dots, g_k\}$ is the output of the algorithm for some input G_1 . Clearly, $\deg(g_i) \leq d$ for all $i \in [k]$. We now prove by contradiction that the output is the d -truncated Gröbner basis of the ideal $\langle G_1 \rangle$ with the grlex ordering. Suppose g is a

16:10 Ideal Membership Problem for Boolean Minority and Dual Discriminator

polynomial of the ideal with $\deg(g) \leq d$, but no $\text{LM}(g_i)$ can divide $\text{LM}(g)$. In fact, since every $g_i \in \langle G_1 \rangle$ we can replace g by $g|_{\{g_1, g_2, \dots, g_k\}}$ (g generalises the reduced S -polynomial). The fact that $g \in \langle G_1 \rangle$ and $g|_{G_1} = 0$ implies that $\text{LM}(g)$ is a linear combination of monomials that are less than $\text{LM}(g)$ (in the grlex order) and hence must be in $B(G_2)$, i.e

$$g|_{G_1} = 0 \implies \text{LM}(g)|_{G_1} = \sum_i k_i b_i|_{G_1}$$

where every $b_i \in B(G_2)$ and $b_i <_{\text{grlex}} \text{LM}(g)$. When the algorithm runs for $q = \text{LM}(g)$, since q was not added to $\text{LM}(G_2)$,

$$\text{LM}(g)|_{G_1} = \sum_j k_j b_j|_{G_1} + f$$

where f is the longest Boolean term of $\text{LM}(g)|_{G_1}$ which does not appear in any previous element of A . But the two equations above imply that $\sum_i k_i b_i|_{G_1} = \sum_j k_j b_j|_{G_1} + f$, which proves that there exists some $b_l \in B(G_2)$ such that a_l has f as its longest Boolean term, so f should have appeared in a_l , a contradiction. Therefore the output is a d -truncated Gröbner basis. Although unnecessary for the IMP_d , we also prove that the output is reduced: every non leading monomial of every polynomial in the output comes from $B(G_2)$ and no leading monomial is a multiple of another by construction. ◀

Thus we have proof of Theorem 1 and Corollary 2.

4 Dual discriminator

We assume in this section that the solution set is non-empty, $D \subset \mathbb{F}$ is any finite domain and the polymorphism in question is the dual discriminator ∇ . The dual discriminator is a majority polymorphism [19, 1] and is often used as a starting point in many CSP-related classifications [1]. For a finite domain D , a ternary operation f is called a majority polymorphism if $f(a, a, b) = f(a, b, a) = f(b, a, a) = a$ for all $a, b \in D$.

► **Definition 14.** *The dual discriminator, denoted by ∇ , is a majority polymorphism such that $\nabla(a, b, c) = a$ for pairwise distinct $a, b, c \in D$.*

The constraints for ∇ -closed problems can be assumed to be binary [19] and are of three types: permutation constraints, complete constraints and two-fan constraints [31, 11]. Bulatov and Rafiey [8] recently proved that the $\text{IMP}(\Gamma)$ over a finite domain is decidable in polynomial time, without showing a proof of membership. They did so by cleverly eliminating the permutation constraints, but were unable to recover a proof for the original problem. We show that a Gröbner basis of the ideal restricted to the permutation constraints can be computed in polynomial time in Section 4.1. We then show in Section 4.2 that the Gröbner basis of constraints that are complete and two-fan constraints can come from a fixed set (see Definition 17). We prove in Section 4.3 that the Gröbner basis of the entire ideal can be found in polynomial time. This Gröbner basis is independent of degree d of the input polynomial: it only contains polynomials with degree less than or equal to $|D|$. Due to space constraints, we give a gist of the proofs as the full proofs will be updated in [4].

4.1 Permutation constraints

A permutation constraint is of the form $R(x_i, x_j)$ where $R = \{(a, \pi_{ij}(a)) \mid a \in D_{ij}\}$ for some $D_{ij} \subseteq D$ and some bijection $\pi_{ij} : D_{ij} \rightarrow D'_{ij}$, where $D'_{ij} \subseteq D$. Let \mathcal{P} be the set of input permutation constraints. We can assume that there exists at most one permutation

constraint over every pair of variables: if there are two on the same set of variables, then their intersection is a permutation constraint. Let $R_{ij}(x_i, x_j)$ represent the unique permutation constraint on variables x_i, x_j , if one exists.

Informally, the goal is to make larger constraints called *chained permutation constraints* (CPC's). Permutation constraints on overlapping variables can be linked to form a larger constraint by using bijections. For example, if there exists $R_{ij}(x_i, x_j), R_{jk}(x_j, x_k) \in \mathcal{P}$, we form a new constraint on x_i, x_j, x_k by using π_{ij} and π_{jk} : the chained permutation constraint is $R(x_i, x_j, x_k)$ where

$$R = \{(\pi_{ij}^{-1}(a), a, \pi_{jk}(a)) \mid a \in D'_{ij} \cap D_{jk}\}.$$

The number of tuples in any CPC is always less than or equal to the domain size, since there is always a bijection between any two variables of a CPC. The constructing of CPC's can be carried out by the arc consistency algorithm described in [21]. A brief working of the algorithm tailored to our application is as follows: let $J \subset [n]$ be an index set for the CPC's (it becomes clear later why there can be at most $\lfloor n/2 \rfloor$ of them but we use n for convenience). We initialise $J = \emptyset$. A general chained permutation constraint CPC_i is defined as $R_i(X_i)$ where R_i is a relation and $X_i \subseteq X$ is a variable set. We keep track of the values that each variable is allowed to take, i.e., S_a is the set of solutions of x_a that satisfies CPC_i for all $x_a \in X_i$. The sets S_a and X_i are updated as CPC_i grows. We define $\sigma_{ab} : S_a \rightarrow S_b$ to be the bijection between any two pairs of variables $x_a, x_b \in X_i$. Hence $\sigma_{ba} = \sigma_{ab}^{-1}$. Let σ_{aa} denote the identity function for all $a \in [n]$. For any permutation constraint R_{pq} in \mathcal{P} , one of the four is true:

- neither x_p nor x_q belong to $\cup_{j \in J} X_j$: in which case we create a CPC. We define $\text{CPC}_i = R_{pq}(x_p, x_q)$ and $X_i = \{x_p, x_q\}$ where $i \in [n] \setminus J$.
- $x_p \in X_i$ and $x_q \notin \cup_{j \in J} X_j$, in which case we expand CPC_i to include R_{pq} and x_q is included in X_i .
- $x_p \in X_i, x_q \in X_j$ and $i \neq j$, in which case CPC_i and CPC_j have a permutation constraint linking two of their variables, so we combine the two CPC's into one. The set $X_i \cup X_j$ is the new X_i and j is deleted from J .
- both $x_p, x_q \in X_i$, in which case we update CPC_i to retain only the common solutions between CPC_i and R_{pq} .

As $R_{pq}(x_p, x_q)$ is now accounted for in some CPC, it is deleted from \mathcal{P} . The algorithm runs until \mathcal{P} is empty. Once \mathcal{P} is empty, CPC_i is defined as

$$\text{CPC}_i := R_i(X_i = \{x_{i_1}, x_{i_2}, \dots, x_{i_r}\}) \text{ where } R_i = \{(a, \sigma_{i_1 i_2}(a), \dots, \sigma_{i_1 i_r}(a)) \mid a \in S_{i_1}\}$$

for each $i \in J$.

► **Remark 15.** For $i \neq j$, $X_i \cap X_j = \emptyset$.

► **Lemma 16.** Let I_{CPC_i} be the combinatorial ideal associated with CPC_i . A Gröbner basis of $\sum_i I_{\text{CPC}_i}$ can be calculated in polynomial time.

The main idea behind the proof is as follows: suppose we see a relation as a matrix where each tuple is a row. Then the arity is equal to the number of columns. The relation R_i in $\text{CPC}_i = R_i(X_i)$ is such that it has at most $|D|!$ pairwise distinct columns, as there exists a bijection between every pair of variables in X_i . If the columns corresponding to x_j and x_k are the same, then the polynomial $x_j - x_k \in I_{\text{CPC}_i}$. We can separate these linear polynomials, and the problem reduces to finding a Gröbner basis of the ideal associated with a constraint that has at most $|D|!$ variables and $|D|$ tuples. This implies that a Gröbner basis can be

computed where the polynomials have degree at most $|D|$. We in fact do not need to find the Gröbner basis of I_{CPC_i} yet. We show in the Section 4.3 as to how we can compute the rest of the polynomials in the Gröbner basis of I_C by using the relations R_i , the sets S_j , the bijections σ_{kl} and polynomials that define complete and two-fan constraints.

4.2 Complete and two-fan constraints

A complete constraint is of the form $R(x_i, x_j)$ where $R = D_i \times D_j$ for some $D_i, D_j \subseteq D$. The polynomials that can represent these constraints are $\prod_{a \in D_i} (x_i - a)$ and $\prod_{b \in D_j} (x_j - b)$. We call these polynomials *partial domain polynomials*. If no such input explicitly exists for a variable, the domain polynomial in that variable itself is the partial domain polynomial. A two-fan constraint is of the form $R(x_i, x_j)$ where $R = (\{a\} \times D_j) \cup (D_i \times \{b\})$ for some $D_i, D_j \subseteq D$ with $a \in D_i, b \in D_j$. This constraint can be represented by the set of polynomials $\{(x_i - a)(x_j - b), \prod_{c \in D_i} (x_i - c), \prod_{d \in D_j} (x_j - d)\}$.

► **Definition 17.** *The set of polynomials \mathcal{D} , \mathcal{F} and \mathcal{L} is defined as follows:*

$$\begin{aligned} \mathcal{D} &= \{\prod_{a \in A} (x_i - a) \mid i \in [n], A \subseteq D\}, \\ \mathcal{F} &= \{(x_i - a)(x_j - b) \mid i, j \in [n], i \neq j\}, \\ \mathcal{L} &= \{x_i - \alpha_2 - (x_j - \beta_2)(\alpha_1 - \alpha_2)/(\beta_1 - \beta_2) \mid i, j \in [n], i \neq j\}, \end{aligned}$$

for all $a, b, \alpha_1, \alpha_2, \beta_1, \beta_2 \in D$ where $\alpha_1 \neq \alpha_2$ and $\beta_1 \neq \beta_2$.

In other words $\mathcal{D} \cup \mathcal{F}$ is the set of all complete constraints and two-fan constraints and \mathcal{L} is the set of polynomials in two variables that pass through two points $(\alpha_1, \beta_1), (\alpha_2, \beta_2) \in D^2$ where $\alpha_1 \neq \alpha_2$ and $\beta_1 \neq \beta_2$. Let $G \subset \mathcal{D} \cup \mathcal{F}$ be the set of polynomials that describes the input complete constraints and two-fan constraints of an instance of $\text{IMP}_d(\Gamma)$. Let $I_{\text{CF}} = \langle G \rangle$ (combinatorial ideal for the Complete and two-Fan constraints). Then

$$I_C = \sum_{i \in J} I_{\text{CPC}_i} + I_{\text{CF}} = \sum_{i \in J} I_{\text{CPC}_i} + \langle G \rangle.$$

► **Lemma 18.** *The reduced Gröbner basis of I_{CF} can be calculated in polynomial time and is a subset of $\mathcal{D} \cup \mathcal{F} \cup \mathcal{L}$.*

Proof sketch. For any pair $f, g \in G$, we show that there are polynomials $H \in \langle G \rangle$ such that $H \in \mathcal{D} \cup \mathcal{F} \cup \mathcal{L}$ and $S(f, g)|_H = 0$. We then include these polynomials in G , i.e. $G := G \cup H$. The cases already considered in Lemma 5.16 of [8] are when:

- $f, g \in \mathcal{F}$ where $f = (x_i - a)(x_j - b)$, $g = (x_i - c)(x_k - d)$ for $a = c$ and $a \neq c$,
- $f \in \mathcal{D}, g \in \mathcal{F}$ where $f = \prod_{a \in D_i} (x_i - a)$, $g = (x_i - c)(x_j - b)$ and $c \in D_i$.

Of the remaining cases, the case that deserves most attention is when $f, g \in \mathcal{F}$ produces a permutation constraint (i.e., when $f = (x_i - a)(x_j - b)$ and $g = (x_i - c)(x_j - d)$ where $a \neq c$ and $b \neq d$).

Hence, the S -polynomial for every two polynomials in G is such that there are polynomials in I_{CF} that belong in $\mathcal{D} \cup \mathcal{F} \cup \mathcal{L}$ which reduce the S -polynomial to zero. In fact, it is not difficult to see that the reduced Gröbner basis is also a subset of $\mathcal{D} \cup \mathcal{F} \cup \mathcal{L}$. Since $|\mathcal{D} \cup \mathcal{F} \cup \mathcal{L}| = O(n^2)$, the reduced Gröbner basis of I_{CF} can be calculated in polynomial time. ◀

■ **Algorithm 2** Calculating Gröbner basis.

Input: G, CPC_i .

Output: Gröbner basis of I_C .

- 1 Compute and replace G by the reduced Gröbner basis of I_{CF} .
- 2 **for** every $g = \prod_{a \in D_p} (x_p - a) \in G \cap \mathcal{D}$ **do**
- 3 **if** $D_p \neq S_p$ **then**
- 4 $S_p := S_p \cap D_p$. Suppose $x_p \in X_i$.
- 5 $S_k := \{\sigma_{pk}(a) \mid a \in S_p\}$ for every $x_k \in X_i \setminus \{x_p\}$.
- 6 Replace g by $\prod_{a \in S_p} (x_p - a)$ in G . Go to Line 1.
- 7 Let $C = G \cap \mathcal{F}$.
- 8 **while** $C \neq \emptyset$ **do**
- 9 Choose $g = (x_p - a)(x_q - b) \in C$. Suppose $x_p \in X_i$.
- 10 **if** $a \notin S_p$ **then**
- 11 Add $x_q - b$ to G if $a \notin S_q$ else add $x_p - a$ to G . Go to Line 1.
- 12 /* At this point $a \in S_p$ and $b \in S_q$. */
- 13 **if** $x_q \in X_j$ for some $i \neq j$ **then**
- 14 **if** $b \notin S_q$ **then**
- 15 Add $x_p - a$ to G . Go to Line 1.
- 16 /* At this point $a \in S_p$ and $b \in S_q$. */
- 17 Let $B := \{(x_k - \sigma_{pk}(a))(x_l - \sigma_{ql}(b)) \mid x_k \in X_i, x_l \in X_j\} \setminus \{g\}$.
- 18 **if** $\exists h \in B$ such that $h|_G \neq 0$ **then**
- 19 $G := G \cup B$. Go to Line 1.
- 20 **if** $x_q \notin \cup_{j \in J} X_j$ **then**
- 21 Let $B := \{(x_k - \sigma_{pk}(a))(x_q - b) \mid x_k \in X_i\}$.
- 22 **if** $\exists h \in B$ such that $h|_G \neq 0$ **then**
- 23 $G := G \cup B$. Go to Line 1.
- 24 Delete g from C .
- 25 Calculate \mathcal{G}_i for every i .
- 26 A Gröbner basis of I_C is $\cup_i \mathcal{G}_i \cup G$.

4.3 Computing a Gröbner basis

► **Theorem 19.** A Gröbner basis of the combinatorial ideal I_C can be calculated in polynomial time.

Proof sketch. Let G be the reduced Gröbner basis of I_{CF} . Then,

$$I_C = \sum_{i \in J} I_{\text{CPC}_i} + I_{\text{CF}} = \sum_{i \in J} \langle \mathcal{G}_i \rangle + \langle G \rangle.$$

For two polynomials $f, g \in \cup_i \mathcal{G}_i \cup G$, we see what the reduced S -polynomial can imply. The straightforward cases are when

- $f, g \in \mathcal{G}_i$: here $S(f, g)|_{\mathcal{G}_i} = 0$ since \mathcal{G}_i is the reduced Gröbner basis of I_{CPC_i} ,
- $f \in \mathcal{G}_i, g \in \mathcal{G}_j$ where $i \neq j$: as f and g don't share any variable in common (see Remark 15), the leading monomials are relatively prime, hence $S(f, g)|_{\{f, g\}} = 0$,
- $f, g \in \mathcal{D} \cup \mathcal{F} \cup \mathcal{L}$: here $S(f, g)|_G = 0$ because of Lemma 18.

The only cases to examine is when $f \in \mathcal{G}_i$ and $g \in G \subset \mathcal{D} \cup \mathcal{F} \cap \mathcal{L}$. In each case, polynomials from $\cup_i \mathcal{G}_i$ and $\mathcal{D} \cup \mathcal{F} \cup \mathcal{L}$ reduce $S(f, g)$ to zero.

Clearly, this Gröbner basis is independent of degree d of the input polynomial. Hence, we have proof of Theorem 3 and Corollary 4. ◀

5 Conclusion

The IMP_d tractability for combinatorial ideals has useful practical applications as it implies bounded coefficients in Sum-of-Squares proofs. A dichotomy result between “hard” (NP-hard) and “easy” (polynomial time) IMPs was achieved for the IMP_0 [6, 34] over the finite domain nearly thirty years after that over the Boolean domain [29]. The IMP_d for $d = O(1)$ over the Boolean domain was tackled by Mastrolilli [23] based on the classification of the IMP through polymorphisms, where the complexity of the IMP_d for five of six polymorphisms was solved. We solve the remaining problem, i.e. the complexity of the $\text{IMP}_d(\Gamma)$ when Γ is closed under the ternary minority polymorphism. This is achieved by showing that the d -truncated reduced Gröbner basis can be computed in polynomial time, thus completing the missing link in the dichotomy result of [23]. We also show that a proof of membership can be found in polynomial time regarding the $\text{IMP}(\Gamma)$ for which constraints are closed under the dual discriminator polymorphism. We believe that generalizing the dichotomy results of solvability of the IMP_d for a finite domain is an interesting and challenging goal that we leave as an open problem.

References

- 1 Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and How to Use Them. In Andrei Krokhin and Stanislav Zivny, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol17.15301.1.
- 2 Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bound on Hilbert’s Nullstellensatz and propositional proofs. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 794–806, 1994.
- 3 Arpitha P. Bharathi and Monaldo Mastrolilli. Ideal Membership Problem and a Majority Polymorphism over the Ternary Domain. In Javier Esparza and Daniel Král, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:13, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.13.
- 4 Arpitha P. Bharathi and Monaldo Mastrolilli. Ideal membership problem for boolean minority, 2020. arXiv:2006.16422.
- 5 Bruno Buchberger. Bruno buchberger’s phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3):475–511, 2006. Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday). doi:10.1016/j.jsc.2005.09.007.
- 6 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs (best paper award). In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330, 2017.
- 7 Andrei A. Bulatov. Constraint satisfaction problems: Complexity and algorithms. *ACM SIGLOG News*, 5(4):4–24, November 2018. doi:10.1145/3292048.3292050.
- 8 Andrei A. Bulatov and Akbar Rafiey. On the complexity of csp-based ideal membership problems, 2020. arXiv:2011.03700.
- 9 Samuel R. Buss and Toniann Pitassi. Good degree bounds on Nullstellensatz refutations of the induction principle. *J. Comput. Syst. Sci.*, 57(2):162–171, 1998.

- 10 Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, 42(1):2:1–2:32, December 2009. doi:10.1145/1592451.1592453.
- 11 Martin C. Cooper, David A. Cohen, and Peter G. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65(2):347–361, 1994. doi:10.1016/0004-3702(94)90021-3.
- 12 David A. Cox, John Little, and Donal O’Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer Publishing Company, Incorporated, 4th edition, 2015.
- 13 Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993. doi:10.1006/jscs.1993.1051.
- 14 Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.
- 15 Dima Grigoriev. Tseitin’s tautologies and lower bounds for Nullstellensatz proofs. In *39th Annual Symposium on Foundations of Computer Science, FOCS ’98, November 8-11, 1998, Palo Alto, California, USA*, pages 648–652, 1998.
- 16 David Hilbert. Ueber die theorie der algebraischen formen. *Mathematische Annalen*, 36:473–534, 1890. doi:10.1007/BF01208503.
- 17 David Hilbert. Ueber die vollen invariantensysteme. *Mathematische Annalen*, 42:313–373, 1893. URL: <http://eudml.org/doc/157652>.
- 18 Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1):185–204, 1998. doi:10.1016/S0304-3975(97)00230-2.
- 19 Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997. doi:10.1145/263867.263489.
- 20 Monique Laurent. *Sums of Squares, Moment Matrices and Optimization Over Polynomials*, pages 157–270. Springer, New York, 2009. doi:10.1007/978-0-387-09686-5_7.
- 21 Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977. doi:10.1016/0004-3702(77)90007-8.
- 22 Monaldo Mastrolilli. The complexity of the ideal membership problem and theta bodies for constrained problems over the boolean domain. *CoRR*, to appear in *ACM Transactions on Algorithms*, abs/1904.04072, 2019. arXiv:1904.04072.
- 23 Monaldo Mastrolilli. The complexity of the ideal membership problem for constrained problems over the boolean domain. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’19*, pages 456–475, Philadelphia, PA, USA, 2019. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=3310435.3310464>.
- 24 Ernst W. Mayr. Membership in polynomial ideals over \mathbb{q} is exponential space complete. In B. Monien and R. Cori, editors, *STACS 89*, pages 400–406, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.
- 25 Ernst W. Mayr and Albert R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46(3):305–329, 1982. doi:10.1016/0001-8708(82)90048-2.
- 26 Ryan O’Donnell. SOS Is Not Obviously Automatizable, Even Approximately. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59:1–59:10, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2017.59.
- 27 Prasad Raghavendra and Benjamin Weitz. On the Bit Complexity of Sum-of-Squares Proofs. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 80:1–80:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2017.80.

- 28 Prasad Raghavendra and Benjamin Weitz. On the bit complexity of sum-of-squares proofs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP, Poland*, pages 80:1–80:13, 2017.
- 29 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, STOC '78*, pages 216–226, New York, NY, USA, 1978. ACM. doi:10.1145/800133.804350.
- 30 Amir Shpilka. Recent results on polynomial identity testing. In Alexander Kulikov and Nikolay Vereshchagin, editors, *Computer Science – Theory and Applications*, pages 397–400, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- 31 Ágnes Szendrei. *Clones in universal algebra*. Les Presses de l'Université de Montréal, 1986.
- 32 Marc R.C. van Dongen. *Constraints, Varieties, and Algorithms*. PhD thesis, Department of Computer Science, University College, Cork, Ireland, 2002. URL: <http://csweb.ucc.ie/~dongen/papers/UCC/02/thesis.pdf>.
- 33 Benjamin Weitz. *Polynomial Proof Systems, Effective Derivations, and their Applications in the Sum-of-Squares Hierarchy*. PhD thesis, EECS Department, University of California, Berkeley, May 2017. URL: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-38.html>.
- 34 Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):30:1–30:78, 2020. doi:10.1145/3402029.

A Ideals, Varieties and Constraints

Let \mathbb{F} denote an arbitrary field (for the applications of this paper $\mathbb{F} = \mathbb{R}$). Let $\mathbb{F}[x_1, \dots, x_n]$ be the ring of polynomials over a field \mathbb{F} and indeterminates x_1, \dots, x_n . Let $\mathbb{F}[x_1, \dots, x_n]_d$ denote the subspace of polynomials of degree at most d .

► **Definition 20.** *The ideal (of $\mathbb{F}[x_1, \dots, x_n]$) generated by a finite set of polynomials $\{f_1, \dots, f_m\}$ in $\mathbb{F}[x_1, \dots, x_n]$ is defined as*

$$\mathbf{I}(f_1, \dots, f_m) \stackrel{\text{def}}{=} \left\{ \sum_{i=1}^m t_i f_i \mid t_1, \dots, t_m \in \mathbb{F}[x_1, \dots, x_n] \right\}.$$

The set of polynomials that vanish in a given set $S \subseteq \mathbb{F}^n$ is called the **vanishing ideal** of S and denoted: $\mathbf{I}(S) \stackrel{\text{def}}{=} \{f \in \mathbb{F}[x_1, \dots, x_n] : f(a_1, \dots, a_n) = 0 \forall (a_1, \dots, a_n) \in S\}$.

► **Definition 21.** *An ideal \mathbf{I} is **radical** if $f^m \in \mathbf{I}$ for some integer $m \geq 1$ implies that $f \in \mathbf{I}$.*

Another common way to denote $\mathbf{I}(f_1, \dots, f_m)$ is by $\langle f_1, \dots, f_m \rangle$ and we will use both notations interchangeably.

► **Definition 22.** *Let $\{f_1, \dots, f_m\}$ be a finite set of polynomials in $\mathbb{F}[x_1, \dots, x_n]$. We call $\mathbf{V}(f_1, \dots, f_m) \stackrel{\text{def}}{=} \{(a_1, \dots, a_n) \in \mathbb{F}^n \mid f_i(a_1, \dots, a_n) = 0 \quad 1 \leq i \leq m\}$ the **affine variety** defined by f_1, \dots, f_m .*

► **Definition 23.** *Let $\mathbf{I} \subseteq \mathbb{F}[x_1, \dots, x_n]$ be an ideal. We will denote by $\mathbf{V}(\mathbf{I})$ the set $\mathbf{V}(\mathbf{I}) = \{(a_1, \dots, a_n) \in \mathbb{F}^n \mid f(a_1, \dots, a_n) = 0 \quad \forall f \in \mathbf{I}\}$.*

► **Theorem 24** ([12], Th.15, p.196). *If \mathbf{I} and \mathbf{J} are ideals in $\mathbb{F}[x_1, \dots, x_n]$, then $\mathbf{V}(\mathbf{I} \cap \mathbf{J}) = \mathbf{V}(\mathbf{I}) \cup \mathbf{V}(\mathbf{J})$.*

A.1 The Ideal-CSP Correspondence

Indeed, let $\mathcal{C} = (X, D, C)$ be an instance of the CSP(Γ) (see Definition 5). Without loss of generality, we shall assume that $D \subset \mathbb{N}$ and $D \subseteq \mathbb{F}$.

Let $Sol(\mathcal{C})$ be the (possibly empty) set of all feasible solutions of \mathcal{C} . In the following, we map $Sol(\mathcal{C})$ to an ideal $I_{\mathcal{C}} \subseteq \mathbb{F}[X]$ such that $Sol(\mathcal{C}) = \mathbf{V}(I_{\mathcal{C}})$.

Let $Y = (x_{i_1}, \dots, x_{i_k})$ be a k -tuple of variables from X and let $R(Y)$ be a non empty constraint from C . In the following, we map $R(Y)$ to a generating system of an ideal such that the projection of the variety of this ideal onto Y is equal to $R(Y)$ (see [32] for more details).

Every $v = (v_1, \dots, v_k) \in R(Y)$ corresponds to some point $v \in \mathbb{F}^k$. It is easy to check [12] that $\mathbf{I}(\{v\}) = \langle x_{i_1} - v_1, \dots, x_{i_k} - v_k \rangle$, where $\langle x_{i_1} - v_1, \dots, x_{i_k} - v_k \rangle \subseteq \mathbb{F}[Y]$ is radical. By Theorem 24, we have

$$R(Y) = \bigcup_{v \in R(Y)} \mathbf{V}(\mathbf{I}(\{v\})) = \mathbf{V}(I_{R(Y)}), \text{ where } I_{R(Y)} = \bigcap_{v \in R(Y)} \mathbf{I}(\{v\}), \quad (7)$$

where $I_{R(Y)} \subseteq \mathbb{F}[Y]$ is zero-dimensional and radical ideal since it is the intersection of radical ideals (see [12], Proposition 16, p.197). Equation (7) states that constraint $R(Y)$ is a variety of \mathbb{F}^k . It is easy to find a generating system for $I_{R(Y)}$:

$$I_{R(Y)} = \langle \prod_{v \in R} (1 - \prod_{j=1}^k \delta_{v_j}(x_{i_j})), \prod_{j \in D} (x_{i_1} - j), \dots, \prod_{j \in D} (x_{i_k} - j) \rangle, \quad (8)$$

where $\delta_{v_j}(x_{i_j})$ are indicator polynomials, i.e. equal to one when $x_{i_j} = v_j$ and zero when $x_{i_j} \in D \setminus \{v_j\}$; polynomials $\prod_{j \in D} (x_{i_k} - j)$ force variables to take values in D and will be denoted as *domain polynomials*.

The smallest ideal (with respect to inclusion) of $\mathbb{F}[X]$ containing $I_{R(Y)} \subseteq \mathbb{F}[\mathbf{x}]$ will be denoted $I_{R(Y)}^{\mathbb{F}[X]}$ and it is called the $\mathbb{F}[X]$ -module of I . The set $Sol(\mathcal{C}) \subset \mathbb{F}^n$ of solutions of $\mathcal{C} = (X, D, C)$ is the intersection of the varieties of the constraints:

$$Sol(\mathcal{C}) = \bigcap_{R(Y) \in C} \mathbf{V}(I_{R(Y)}^{\mathbb{F}[X]}) = \mathbf{V}(I_{\mathcal{C}}), \text{ where } I_{\mathcal{C}} = \sum_{R(Y) \in C} I_{R(Y)}^{\mathbb{F}[X]}. \quad (9)$$

The following properties follow from Hilbert's Nullstellensatz.

► **Theorem 25.** *Let \mathcal{C} be an instance of the CSP(Γ) and $I_{\mathcal{C}}$ defined as in (9). Then*

$$(Weak Nullstellensatz) \quad \mathbf{V}(I_{\mathcal{C}}) = \emptyset \Leftrightarrow 1 \in I_{\mathcal{C}} \Leftrightarrow I_{\mathcal{C}} = \mathbb{F}[X], \quad (10)$$

$$(Strong Nullstellensatz) \quad \mathbf{I}(\mathbf{V}(I_{\mathcal{C}})) = \sqrt{I_{\mathcal{C}}}, \quad (11)$$

$$(Radical Ideal) \quad \sqrt{I_{\mathcal{C}}} = I_{\mathcal{C}}. \quad (12)$$

Theorem 25 follows from a simple application of the celebrated and basic result in algebraic geometry known as Hilbert's Nullstellensatz. In the general version of Nullstellensatz it is necessary to work in an algebraically closed field and take a radical of the ideal of polynomials. In our special case it is not needed due to the presence of domain polynomials. Indeed, the latter implies that we know a priori that the solutions must be in \mathbb{F} (note that we are assuming $D \subseteq \mathbb{F}$).

A.2 Gröbner bases

In this section we suppose a fixed monomial ordering $>$ on $\mathbb{F}[x_1, \dots, x_n]$ (see [12], Definition 1, p.55), which will not be defined explicitly. We can reconstruct the monomial $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ from the n -tuple of exponents $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n$. This establishes a one-to-one correspondence between the monomials in $\mathbb{F}[x_1, \dots, x_n]$ and $\mathbb{Z}_{\geq 0}^n$. Any ordering $>$ we establish on the space $\mathbb{Z}_{\geq 0}^n$ will give us an ordering on monomials: if $\alpha > \beta$ according to this ordering, we will also say that $x^\alpha > x^\beta$. The two monomial orderings that we use in this paper are the lexicographic order $>_{\text{lex}}$ and the graded lexicographic ordering $>_{\text{grlex}}$.

► **Definition 26.** Let $\alpha = (\alpha_1, \dots, \alpha_n), \beta = (\beta_1, \dots, \beta_n) \in \mathbb{Z}_{\geq 0}^n$ and $|\alpha| = \sum_{i=1}^n \alpha_i, |\beta| = \sum_{i=1}^n \beta_i$.

- (i) We say $\alpha >_{\text{lex}} \beta$ if, in the vector difference $\alpha - \beta \in \mathbb{Z}^n$, the left most nonzero entry is positive. We will write $x^\alpha >_{\text{lex}} x^\beta$ if $\alpha >_{\text{lex}} \beta$.
- (ii) We say $\alpha >_{\text{grlex}} \beta$ if $|\alpha| > |\beta|$, or $|\alpha| = |\beta|$ and $\alpha >_{\text{lex}} \beta$.

► **Definition 27.** For any $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n$ let $x^\alpha \stackrel{\text{def}}{=} \prod_{i=1}^n x_i^{\alpha_i}$. Let $f = \sum_{\alpha} a_{\alpha} x^{\alpha}$ be a nonzero polynomial in $\mathbb{F}[x_1, \dots, x_n]$ and let $>$ be a monomial order.

- (i) The **multidegree** of f is $\text{multideg}(f) \stackrel{\text{def}}{=} \max(\alpha \in \mathbb{Z}_{\geq 0}^n : a_{\alpha} \neq 0)$.
- (ii) The **degree** of f is $\text{deg}(f) = |\text{multideg}(f)|$. In this paper, this is always according to grlex order.
- (iii) The **leading coefficient** of f is $\text{LC}(f) \stackrel{\text{def}}{=} a_{\text{multideg}(f)} \in \mathbb{F}$.
- (iv) The **leading monomial** of f is $\text{LM}(f) \stackrel{\text{def}}{=} x^{\text{multideg}(f)}$ (with coefficient 1).
- (v) The **leading term** of f is $\text{LT}(f) \stackrel{\text{def}}{=} \text{LC}(f) \cdot \text{LM}(f)$.

The concept of *reduction*, also called *multivariate division* or *normal form computation*, is central to Gröbner basis theory. It is a multivariate generalization of the Euclidean division of univariate polynomials.

► **Definition 28.** Fix a monomial order and let $G = \{g_1, \dots, g_t\} \subset \mathbb{F}[x_1, \dots, x_n]$. Given $f \in \mathbb{F}[x_1, \dots, x_n]$, we say that **f reduces to r modulo G** , written $f \rightarrow_G r$, if f can be written in the form $f = A_1 g_1 + \cdots + A_t g_t + r$ for some $A_1, \dots, A_t, r \in \mathbb{F}[x_1, \dots, x_n]$, such that:

- (i) No term of r is divisible by any of $\text{LT}(g_1), \dots, \text{LT}(g_t)$.
- (ii) Whenever $A_i g_i \neq 0$, we have $\text{multideg}(f) \geq \text{multideg}(A_i g_i)$.

The polynomial remainder r is called a **normal form of f by G** and will be denoted by $f|_G$.

A normal form of f by G , i.e. $f|_G$, can be obtained by repeatedly performing the following until it cannot be further applied: choose any $g \in G$ such that $\text{LT}(g)$ divides some term t of f and replace f with $f - \frac{t}{\text{LT}(g)}g$. Note that the order we choose the polynomials g in the division process is not specified.

In general a normal form $f|_G$ is not uniquely defined. Even when f belongs to the ideal generated by G , i.e. $f \in \mathbf{I}(G)$, it is not always true that $f|_G = 0$.

► **Example 29.** Let $f = xy^2 - y^3$ and $G = \{g_1, g_2\}$, where $g_1 = xy - 1$ and $g_2 = y^2 - 1$. Consider the graded lexicographic order (with $x > y$) and note that $f = y \cdot g_1 - y \cdot g_2 + 0$ and $f = 0 \cdot g_1 + (x - y) \cdot g_2 + x - y$.

This non-uniqueness is the starting point of Gröbner basis theory.

► **Definition 30.** Fix a monomial order on the polynomial ring $\mathbb{F}[x_1, \dots, x_n]$. A finite subset $G = \{g_1, \dots, g_t\}$ of an ideal $I \subseteq \mathbb{F}[x_1, \dots, x_n]$ different from $\{0\}$ is said to be a **Gröbner basis** (or **standard basis**) if $\langle \text{LT}(g_1), \dots, \text{LT}(g_t) \rangle = \langle \text{LT}(I) \rangle$, where we denote by $\langle \text{LT}(I) \rangle$ the ideal generated by the elements of the set $\text{LT}(I)$ of leading terms of nonzero elements of I .

► **Definition 31.** A **reduced Gröbner basis** for a polynomial ideal I is a Gröbner basis G for I such that:

- (i) $\text{LC}(g) = 1$ for all $g \in G$.
- (ii) For all $g \in G$, g cannot reduce any other polynomial from G , i.e. $f|_g = f$ for every $f \in G \setminus \{g\}$.

It is known (see [12], Theorem 5, p.93) that for a given monomial ordering, a polynomial ideal $I \neq \{0\}$ has a reduced Gröbner basis (see Definition 31), and the reduced Gröbner basis is unique.

► **Proposition 32** ([12], Proposition 1, p.83). Let $I \subset \mathbb{F}[x_1, \dots, x_n]$ be an ideal and let $G = \{g_1, \dots, g_t\}$ be a Gröbner basis for I . Then given $f \in \mathbb{F}[x_1, \dots, x_n]$, f can be written in the form $f = A_1g_1 + \dots + A_tg_t + r$ for some $A_1, \dots, A_t, r \in \mathbb{F}[x_1, \dots, x_n]$, such that:

- (i) No term of r is divisible by any of $\text{LT}(g_1), \dots, \text{LT}(g_t)$.
- (ii) Whenever $A_i g_i \neq 0$, we have $\text{multideg}(f) \geq \text{multideg}(A_i g_i)$.
- (iii) There is a unique $r \in \mathbb{F}[x_1, \dots, x_n]$.

In particular, r is the remainder on division of f by G no matter how the elements of G are listed when using the division algorithm.

► **Corollary 33** ([12], Corollary 2, p.84). Let $G = \{g_1, \dots, g_t\}$ be a Gröbner basis for $I \subseteq \mathbb{F}[x_1, \dots, x_n]$ and let $f \in \mathbb{F}[x_1, \dots, x_n]$. Then $f \in I$ if and only if the remainder on division of f by G is zero.

► **Definition 34.** We will write $f|_F$ for the remainder of f by the ordered s -tuple $F = (f_1, \dots, f_s)$. If F is a Gröbner basis for $\langle f_1, \dots, f_s \rangle$, then we can regard F as a set (without any particular order) by Proposition 32.

The “obstruction” to $\{g_1, \dots, g_t\}$ being a Gröbner basis is the possible occurrence of polynomial combinations of the g_i whose leading terms are not in the ideal generated by the $\text{LT}(g_i)$. One way (actually the only way) this can occur is if the leading terms in a suitable combination cancel, leaving only smaller terms. The latter is fully captured by the so called S -polynomials that play a fundamental role in Gröbner basis theory.

► **Definition 35.** Let $f, g \in \mathbb{F}[x_1, \dots, x_n]$ be nonzero polynomials. If $\text{multideg}(f) = \alpha$ and $\text{multideg}(g) = \beta$, then let $\gamma = (\gamma_1, \dots, \gamma_n)$, where $\gamma_i = \max(\alpha_i, \beta_i)$ for each i . We call x^γ the **least common multiple** of $\text{LM}(f)$ and $\text{LM}(g)$, written $x^\gamma = \text{lcm}(\text{LM}(f), \text{LM}(g))$. The **S -polynomial** of f and g is the combination $S(f, g) = \frac{x^\gamma}{\text{LT}(f)} \cdot f - \frac{x^\gamma}{\text{LT}(g)} \cdot g$.

The use of S -polynomials to eliminate leading terms of multivariate polynomials generalizes the row reduction algorithm for systems of linear equations. If we take a system of homogeneous linear equations (i.e.: the constant coefficient equals zero), then it is not hard to see that bringing the system in triangular form yields a Gröbner basis for the system.

► **Theorem 36 (Buchberger’s Criterion).** (See e.g. [12], Theorem 3, p.105) A basis $G = \{g_1, \dots, g_t\}$ for an ideal I is a Gröbner basis if and only if $S(g_i, g_j) \rightarrow_G 0$ for all $i \neq j$.

By Theorem 36 it is easy to show whether a given basis is a Gröbner basis. Indeed, if G is a Gröbner basis then given $f \in \mathbb{F}[x_1, \dots, x_n]$, $f|_G$ is unique and it is the remainder on division of f by G , no matter how the elements of G are listed when using the division algorithm.

■ **Algorithm 3** Buchberger's Algorithm.

```

1: Input: A finite set  $F = \{f_1, \dots, f_s\}$  of polynomials
2: Output: A finite Gröbner basis  $G$  for  $\langle f_1, \dots, f_s \rangle$ 
3:  $G := F$ 
4:  $C := G \times G$ 
5: while  $C \neq \emptyset$  do
6:   Choose a pair  $(f, g) \in C$ 
7:    $C := C \setminus \{(f, g)\}$ 
8:    $h := S(f, g)|_G$ 
9:   if  $h \neq 0$  then
10:     $C := C \cup (G \times \{h\})$ 
11:     $G := G \cup \{h\}$ 
12:   end if
13: end while
14: Return  $G$ 

```

Furthermore, Theorem 36 leads naturally to an algorithm for computing Gröbner bases for a given ideal $I = \langle f_1, \dots, f_s \rangle$: start with a basis $G = \{f_1, \dots, f_s\}$ and for any pair $f, g \in G$ with $S(f, g)|_G \neq 0$ add $S(f, g)|_G$ to G . This is known as Buchberger's algorithm [5] (for more details see Algorithm 3 in Section A.2.1).

Note that Algorithm 3 is non-deterministic and the resulting Gröbner basis is not uniquely determined by the input. This is because the normal form $S(f, g)|_G$ (see Algorithm 3, line 8) is not unique as already remarked. We observe that one simple way to obtain a deterministic algorithm (see [12], Theorem 2, p. 91) is to replace $h := S(f, g)|_G$ in line 8 with $h := S(f, g)|_G$ (see Definition 34), where in the latter G is an ordered tuple. However, this is potentially dangerous and inefficient. Indeed, there are simple cases where the combinatorial growth of set G in Algorithm 3 is out of control very soon.

A.2.1 Construction of Gröbner Bases

Buchberger's algorithm [5] can be formulated as in Algorithm 3. The pairs that get placed in the set C are often referred to as *critical pairs*. Every newly added reduced S -polynomial enlarges the set C . If we use $h := S(f, g)|_G$ in line 8 then there are simple cases where the situation is out of control. This combinatorial growth can be controlled to some extent by eliminating unnecessary critical pairs.

Graph Traversals as Universal Constructions

Siddharth Bhaskar  

Department of Computer Science, University of Copenhagen, Denmark

Robin Kaarsgaard   

School of Informatics, University of Edinburgh, UK

Abstract

We exploit a decomposition of graph traversals to give a novel characterization of depth-first and breadth-first traversals by means of universal constructions. Specifically, we introduce functors from two different categories of *edge-ordered* directed graphs into two different categories of transitively closed edge-ordered graphs; one defines the lexicographic depth-first traversal and the other the lexicographic breadth-first traversal. We show that each functor factors as a composition of universal constructions, and that the usual presentation of traversals as linear orders on vertices can be recovered with the addition of an inclusion functor. Finally, we raise the question of to what extent we can recover search algorithms from the categorical description of the traversal they compute.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Theory of computation → Models of computation

Keywords and phrases graph traversals, adjunctions, universal constructions, category theory

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.17

Related Version *Full Version*: <https://arxiv.org/abs/2104.14877> [4]

Funding *Siddharth Bhaskar*: Supported by DFF | *Natural Sciences* research project 1 fellowship 115554 *Cons-Free Recursion Theory*.

Robin Kaarsgaard: Supported by DFF | *Natural Sciences* international postdoctoral fellowship 0131-00025B *Landauer Meets von Neumann: Reversibility in Categorical Quantum Semantics*.

Acknowledgements We would like to thank Steve Lindell and Scott Weinstein for inspiring us to study traversals in terms of their predecessor functions, as well as suggesting the characterization of breadth-first traversals in Corollary 27. We also thank Jade Master for discussions relating to this work. We are indebted to the anonymous reviewers for their thorough and detailed comments.

1 Introduction

Graph searches are algorithms for visiting the vertices in a connected graph from a prescribed source. Both graph searches and their resulting vertex orders, or *traversals*, are absolutely fundamental in the theory of graph algorithms, and have important applications in other areas of theoretical computer science such as computational complexity theory.

We start from the premise that a concept as natural as a traversal should be obtainable canonically from the original graph. For example, let us consider the graph G in Figure 1, and fix a as a source. Of the six vertex orderings of G starting with a , two are not traversals: (a, d, b, c) and (a, d, c, b) . This is because while searching a graph, each vertex added must be in the boundary of previously visited vertices, but d is not in the boundary of $\{a\}$.

Of the four remaining vertex orders, two are breadth-first traversals $((a, b, c, d)$ and $(a, c, b, d))$ and two are depth-first traversals $((a, b, d, c)$ and $(a, c, d, b))$. This can easily be checked by hand: in a breadth-first search, we go level-by-level, and must visit both b and c before we visit d . In a depth-first search, we prioritize the neighbor of the most recently visited vertex, so we visit d before the latter of $\{b, c\}$.



© Siddharth Bhaskar and Robin Kaarsgaard;
licensed under Creative Commons License CC-BY 4.0

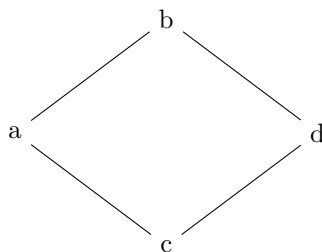
46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 17; pp. 17:1–17:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** A 4-cycle.

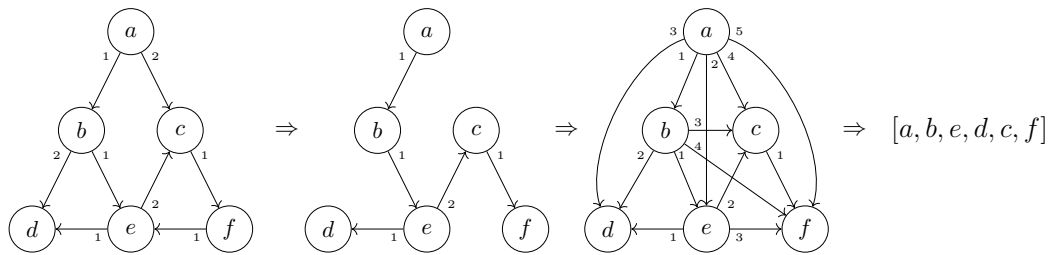
However, notice that there is no way of canonically distinguishing between the two breadth-first or the two depth-first traversals. Concretely, once we visit a , there is no canonical way to choose between b and c . A natural fix is to linearly order each neighborhood and visit lesser neighbors first. We call the resulting traversals *lexicographic*. For example, if we say that $b < c$, then the lexicographic breadth-first traversal is (a, b, c, d) and the lexicographic depth-first traversal is (a, b, d, c) . If we say that $c < b$, we get (a, c, b, d) and (a, c, d, b) respectively. We call a graph whose neighborhoods are linearly ordered an *edge-ordered* graph.

In the present paper, we show that both the lexicographic breadth-first traversal and lexicographic depth-first traversal are canonically obtainable from a given edge-ordered graph with a distinguished source. Specifically, we equip edge-ordered graphs with two different kinds of morphisms, and obtain lexicographic breadth- and depth-first traversals by applying a functor out of each category of edge-ordered graphs into the category of linear orders. We furthermore factor each functor as a composition of a forgetful functor and a sequence of *universal* (free and cofree) constructions on edge-ordered graphs.

At a first approximation, each lexicographic traversal can be expressed as the composition of a least-path tree and a transitive closure (see Figure 2). This decomposition was first observed in [7] – not in the context of category theory – where it was used to derive efficient parallel algorithms; see also [3, 6]. The main technical contribution of our paper is in identifying precisely the right notions of edge-ordered graphs and homomorphisms that allow us to formulate least-path trees and transitive closures as universal constructions.

To the best of our knowledge, equipping algorithmic problems with a categorical structure is a relatively recent idea. While graphs have been studied extensively from a categorical point of view, the focus has been on topics such as graph rewriting and string diagrams [8, 10] and relationships with properads [9] rather than graph algorithms. Closer to our approach, a surprisingly simple and elegant characterization of reachability in all *coalgebras* (i.e., including graphs) is studied in [2, 14]. A categorical treatment of the open algebraic path problem is given in [11], and a compositional algorithm for reachability in Petri nets is described in [12].

In [1], Abramsky describes a “great divide” between those areas of theoretical computer science focused on *structure* (semantics and type theory), and those focused on *power* (computability and complexity theory); they have “almost disjoint communities” with “no common technical language or tools.” He proposes a high-level “theory-building” program of integrating these approaches with the intent of solving hard problems, akin to Grothendieck’s program in algebraic geometry. We envision developing a theory of *compositional graph algorithms through universal properties* as a step along this way. A long-term goal of such a project would be to see whether one could recover *algorithms* from *problem statements*, if the latter are suitably formulated.



■ **Figure 2** The construction of the lexicographic depth-first traversal starting from a on a given edge-ordered graph. First we extract the least-path tree, transitively close it, then isolate the ordered neighborhood of a . Numerals indicate the edge ordering. Each transformation is universal, except for a “silent” (forgetful) transformation fixing the transitive graph but forgetting some structure on morphisms.

This paper is structured as follows: We describe the necessary background on graphs and (lexicographic) traversals in Section 2, and present an alternate formulation of traversals in Section 3 that the remaining work will build on. We go on to describe the categories of edge-ordered graphs on which our work is founded in Section 4, and present the two categories of least path trees and universal constructions associated with lexicographic breadth-first and depth-first traversals respectively in Section 5 and Section 6. The categories of transitively closed least path trees and their universal constructions are presented in Section 7 and Section 8. Finally, we summarise our categorical construction of lexicographic breadth-first and depth-first traversals by universal means in Section 9, and end with some concluding remarks in Section 10.

Note that in the interests of space, we omit proofs in the pre-categorical development (Sections 2 and 3), and we relegate the category-theoretic proofs from Sections 4 through Section 9 to the Appendix. The omitted proofs may be found in [4].

2 Background and Preliminaries

Our objective is to give a categorical formulation of the canonical breadth-first and depth-first traversals of directed edge-ordered graphs. But first, in Sections 2 and 3, we give a “pre-categorical” account of these traversals in terms of extremal-path trees, which motivates the subsequent categorical treatment in Sections 4 through 8. The main results of Sections 2 and 3 can be found in [7], but we provide our own presentation.

► **Definition 1.** A (directed) graph is a pair (V, \rightarrow) where V is the set of vertices and $\rightarrow \subseteq V \times V$ is the edge relation.

► **Definition 2.** The neighborhood of a vertex $v \in V$, denoted $N(v)$, is the set of outgoing edges of v ; that is, $N(v) = \rightarrow \cap \{(v, x) \mid x \in V\}$. A graph with a distinguished vertex is pointed.

► **Definition 3.** A path in a graph is a finite sequence of vertices $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$; say that v_1 is the source of the path, and v_n the target. We will write $u \rightsquigarrow v$ to say that there is a path from u to v , and $u \xrightarrow{\pi} v$ about a specific such path named π . We say that a path is proper when no vertex occurs more than once in the path, i.e., when $v_i = v_j$ implies $i = j$ for all $i, j \in I$. We use ε to denote the unique empty (proper) path in each neighborhood.

► **Remark 4.** Notice that $u \rightsquigarrow v$ iff there exists a proper path from u to v , as repetitions in a path can always be deleted.

► **Definition 5.** Two paths are co-initial in case they share a source, and co-final in case they share a target. If the source of σ agrees with the target of π , we can compose them to obtain $\pi\sigma$. For two co-initial paths π and σ , we say $\pi \sqsubset \sigma$ (read: σ extends π) in case π is a proper initial subsequence (i.e., a proper prefix) of σ .

► **Definition 6.** A pointed graph (G, v_0) is connected if for every vertex v , there exists a path $v_0 \rightsquigarrow v$.

2.1 Graph searching and traversals

By a *graph search*, we mean an algorithm for visiting all the vertices in a connected graph, starting at a given source. Two of the most important types of graph search are *depth-first* and *breadth-first*:

► **Definition 7 (Depth-first search).** Given as input a finite connected graph $G = (V, E)$, initialize a list $L = ()$, and a stack $S = (v_0)$ for some vertex v_0 . While S is nonempty, pop the first element v from S . If v is already contained in L , go back to the start of the loop. Otherwise, let $L = (L, v)$, and push every vertex in ∂v onto S , where $\partial v = N(v) \setminus L$.

► **Definition 8 (Breadth-first search).** Given as input a finite connected graph $G = (V, E)$, initialize a list $L = ()$, and a queue $Q = (v_0)$ for some vertex v_0 . While S is nonempty, dequeue the front element v from Q . If v is already contained in L , go back to the start of the loop. Otherwise, let $L = (L, v)$, and enqueue every vertex in ∂v to the back of Q , where $\partial v = N(v) \setminus L$.

Note that depth-first and breadth-first search are nondeterministic, in the sense that we do not specify which order to add vertices from ∂v in. Moreover, vertices may occur more than once in S or Q , as the same vertex may be added as a neighbor multiple times. However, vertices may not occur more than once in L ; when depth-first or breadth-first search is complete, L lists all the vertices in G .

As we vary over all the nondeterministic traces of depth-first search or breadth-first search over a graph G , the resulting orderings are depth-first or breadth-first traversals.

► **Definition 9.** A linear ordering $<$ of the vertices of a graph G is a depth-first traversal, respectively breadth-first traversal of G if there exists some computation of depth-first search, respectively breadth-first search, on G according to which vertices are added to L in exactly the order $<$.

Both depth- and breadth-first traversals have important characterizations in the first-order language of ordered graphs; these are due to Corneil and Krueger [5], who state them in the special case of undirected graphs. Here, we only need to know that they are necessary.

► **Lemma 10.** Suppose G is a finite, connected graph and $\cdot < \cdot$ is a depth-first traversal of G . Then for any vertices $u < v < w$ such that $u \rightarrow w$, there exists some v' such that $u \leq v' < v$ and $v' \rightarrow v$.

► **Lemma 11.** Suppose G is a finite, connected graph and $\cdot < \cdot$ is a breadth-first traversal of G . Then for any vertices $u < v < w$ such that $u \rightarrow w$, there exists some v' such that $v' \leq u < v$ and $v' \rightarrow v$.

These characterizations motivate the following definitions.

► **Definition 12.** Let G be a finite connected graph and $\cdot < \cdot$ be a depth-first traversal of G . Then for any vertex v , define the depth-first predecessor $dfp(v)$ to be the greatest $u < v$ such that $u \rightarrow v$, if v is not the minimal vertex. For the minimal vertex v , let $dfp(v) = v$.

► **Definition 13.** Let G be a finite connected graph and $\cdot < \cdot$ be a breadth-first traversal of G . Then for any vertex v , define the breadth-first predecessor $bfp(v)$ to be the least $u < v$ such that $u \rightarrow v$, if v is not the minimal vertex. For the minimal vertex v , let $bfp(v) = v$.

Depth- and breadth-first predecessors allow for elegant restatements of Lemmata 10 and 11: if $\cdot < \cdot$ is a depth-first traversal of G , then for any $v < w$, if $dfp(w) < v$, then $dfp(w) \leq dfp(v)$. Similarly, if $\cdot < \cdot$ is a breadth-first traversal of G then for any $v < w$, if $bfp(w) < v$, then $bfp(v) \leq bfp(w)$. In fact the second condition is equivalent to saying that bfp is weakly monotone, viz., $v \leq w \implies bfp(v) \leq bfp(w)$.

For the next definition, notice that the orbits $\{v, dfp(v), dfp^2(v), \dots\}$ and $\{v, bfp(v), bfp^2(v), \dots\}$ of any vertex v are finite, and their *reversals* are paths from the least vertex v_0 to v .

► **Definition 14.** Let G be a finite connected graph, $\cdot < \cdot$ be a depth-first traversal of G , and v_0 be the $<$ -least vertex. Then the canonical df-path from v_0 to any vertex v is the path $v_0 \rightarrow v_1 \rightarrow \dots \rightarrow v_{\ell-1}$ such that $v_{\ell-1} = v$, and for every $0 \leq i < \ell - 1$, $dfp(v_{i+1}) = v_i$.

Analogously, the canonical bf-path from v_0 to any vertex v is the path $v_0 \rightarrow v_1 \rightarrow \dots \rightarrow v_{\ell-1}$ such that $v_{\ell-1} = v$, and for every $0 \leq i < \ell - 1$, $bfp(v_{i+1}) = v_i$.

2.2 Lexicographic searching

The objective of this paper is to show that the depth-first and breadth-first traversals of a graph is *canonical* in some precise categorical sense. Of course, this cannot be true at face value: in a complete graph, every ordering of the vertices is both a breadth-first and a depth-first traversal, and none of them is canonical.

As we remarked in the introduction, an edge-ordering is precisely the amount of additional structure we need. Over such graphs, we can make searching deterministic, as we now show.

► **Definition 15.** A finite edge-ordered graph is a finite graph where each neighborhood is equipped with a (strict) linear order $\triangleleft \cdot$.

► **Definition 16** (Lexicographic depth-first search). Let G be a finite, pointed, connected, edge-ordered graph with distinguished vertex v_0 . Initialize a list $L = ()$ and a stack $S = (v_0)$. While S is nonempty, pop the first element v from S . If v is already contained in L , go back to the start of the loop. Otherwise, let $L = (L, v)$, and push every vertex in ∂v onto S in reverse \triangleleft -order, where ∂v is the set of neighbors of v not in L .

► **Remark 17.** We push vertices from ∂v onto S in reverse \triangleleft -order, so that the least vertices from ∂v end up on top of S .

► **Definition 18** (Lexicographic breadth-first search). Let G be a finite, pointed, connected, edge-ordered graph with distinguished vertex v_0 . Initialize a list $L = ()$ and a queue $Q = (v_0)$. While Q is nonempty, dequeue first element v from S . If v is already contained in L , go back to the start of the loop. Otherwise, let $L = (L, v)$, and enqueue every vertex from ∂v onto Q in \triangleleft -order, where ∂v is the set of neighbors of v not in L .

► **Definition 19.** The depth-first traversal computed by lexicographic depth-first search on a finite, pointed, connected, edge-ordered graph G , is its lexicographic depth-first traversal. Similarly, the breadth-first traversal computed by lexicographic breadth-first search is its lexicographic breadth-first traversal.

► Remark 20. The usage of *lexicographic (depth, breadth)-first (search, traversal)* is ambiguous in the literature. Here, we use it in the same way as Delatorre and Kruskal [7, 6]; however, the *lexicographic breadth-first search* of Rose and Tarjan [13] and the analogous *depth-first* version of Corneil and Krueger [5] are different. The latter are further refinements of breadth-first search and depth-first search over graphs, not a “determinization” by an edge-ordering.

3 Orders on paths

In this section, we give different characterizations of the lexicographic depth-first and breadth-first traversals, independent of any graph search, which suggest the category-theoretic treatment that occupies us for the rest of the paper.

► **Definition 21.** Fix an edge-ordered graph G . Define the lexicographic path relation $\cdot \prec \cdot$ on any proper co-initial paths π and σ in G as follows:

- (i) if $\pi \sqsubset \sigma$, then $\pi \prec \sigma$, similarly if $\sigma \sqsubset \pi$, then $\sigma \prec \pi$; otherwise,
- (ii) let ζ be the longest common prefix of π and σ , let u be the target of ζ , and let v_1 and v_2 be the first vertices immediately following ζ in π and σ respectively. Order $\pi \prec \sigma \iff u \rightarrow v_1 \triangleleft u \rightarrow v_2$.

► Remark 22. The following properties hold of \prec :

1. The empty path (v) is least among all paths from v .
2. If $\pi \prec \sigma$ and $\pi \not\sqsubseteq \sigma$, then for any α and β , $\pi\alpha \prec \sigma\beta$.
3. If $\pi \prec \sigma$, then $\alpha\pi \prec \alpha\sigma$.
4. If $\alpha\pi \prec \alpha\sigma$, then $\pi \prec \sigma$.

Since the set of proper paths is finite, any subset has a \prec -least element, which justifies the next definitions.

► **Definition 23.** For vertices u, v in a finite edge-ordered graph, let $\min(u \rightsquigarrow v)$ be the lexicographically least proper path from u to v .

► **Definition 24.** For vertices u, v in a finite edge-ordered graph, let $\min^s(u \rightsquigarrow v)$ be the lexicographically least shortest path from u to v .

In fact, it will be convenient to define the following relation, the *shortlex* order:

► **Definition 25.** Let $\pi \prec^s \sigma$ mean that either $|\pi| < |\sigma|$, or $|\pi| = |\sigma|$ and $\pi \prec \sigma$.

In this case, $\min^s(u \rightsquigarrow v)$ is simply, the \prec^s -least path $u \rightsquigarrow v$.

In the remainder of this section, we fix a finite, pointed, connected, edge-ordered graph G with distinguished element v_0 . Reserve the symbol $\cdot \triangleleft \cdot$ for the given ordering on co-initial edges and $\cdot \prec \cdot$ for the induced lexicographic ordering on co-initial paths. Let $\cdot <_D \cdot$ and $\cdot <_B \cdot$ denote the lexicographic depth-first and breadth-first traversals respectively. Let $\mathfrak{P}v$ and $\mathfrak{Q}v$ denote the canonical df- and bf-paths from v_0 to v with respect to $\cdot <_D \cdot$ and $\cdot <_B \cdot$ respectively.

Our goal is to prove that $<_D$ and $<_B$ satisfy the following properties,

$$u <_D v \iff \mathfrak{P}u \prec \mathfrak{P}v$$

$$u <_B v \iff \mathfrak{Q}u \prec^s \mathfrak{Q}v,$$

which then in turn yield the following alternate characterizations:

$$u <_D v \iff \min(v_0 \rightsquigarrow v) \prec \min(v_0 \rightsquigarrow u)$$

$$u <_B v \iff \min^s(v_0 \rightsquigarrow v) \prec \min^s(v_0 \rightsquigarrow u).$$

► **Lemma 26.** If $u \leq_B v$, then for any path $\pi : v_0 \rightsquigarrow v$, $|\mathfrak{Q}u| \leq |\pi|$. In addition, if $|\mathfrak{Q}u| = |\pi|$, then $\mathfrak{Q}u \preceq \pi$.

► **Corollary 27.** *The following are all true of the operator Ω :*

1. *For any vertex v of G , $\Omega v = \min^s(v_0 \rightsquigarrow v)$.*
2. *$u \leq_B v$ iff $\Omega u \preceq^s \Omega v$.*

Corollary 27 yields $u <_B v \iff \min^s(v_0 \rightsquigarrow v) < \min^s(v_0 \rightsquigarrow u)$, as desired.

► **Definition 28.** *Define $v <_D w$ in case $\min(v_0 \rightsquigarrow v) < \min(v_0 \rightsquigarrow w)$.*

► **Lemma 29.** *For any vertex v of G , $\mathfrak{F}v = \min(v_0 \rightsquigarrow v)$.*

► **Theorem 30.** *The order $\cdot <_D \cdot$ is exactly the lexicographic depth-first traversal $\cdot <_D \cdot$ of G .*

4 Categories of graphs

We now work towards a categorical formulation of the above material. This means we need to categorify, i.e., equip with morphisms, all the objects we defined in Section 2, as well as introduce some new objects.

Continuing the convention established above, we use the symbol $<$ and variations thereof to refer to orderings of co-initial paths, and the symbol \triangleleft to refer to neighborhood orders, i.e., orderings of co-initial edges. We reserve the symbol $<$ for vertex orders, but these will not reappear until Section 9.

► **Definition 31.** *A homomorphism of graphs $G \xrightarrow{h} H$ is a function from G -vertices to H -vertices which preserves edge connectivity: For all G -vertices u, v , $u \rightarrow v$ in G implies $h(u) \rightarrow h(v)$ in H . A homomorphism of pointed graphs $G \xrightarrow{h} H$ must additionally map the distinguished vertex of G , and only that vertex, to the distinguished vertex of H .*

► **Definition 32.** *If $h : G \rightarrow H$ is a graph homomorphism and π is the path $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$ in G , then $h(\pi)$ is defined to be the path $h(v_1) \rightarrow h(v_2) \rightarrow \dots \rightarrow h(v_n)$ in H .*

► **Remark 33.** The following hold of any homomorphism $h : G \rightarrow H$;

1. $h(\varepsilon) = \varepsilon$, $h(\pi\sigma) = h(\pi)h(\sigma)$, and if $\pi \sqsubset \sigma$, then $h(\pi) \sqsubset h(\sigma)$.
2. If h is injective on vertices and π is a proper path, then $h(\pi)$ is a proper path.
3. If h is injective on vertices, then h preserves longest common prefixes.

We now define homomorphisms of edge-ordered graphs (cf. Definition 15). In addition to the “straightforward” property of being monotone on neighborhoods, we also want to consider homomorphisms that preserve lexicographically least paths. This gives us two refinements of the notion of homomorphism, depending on whether we want to preserve lexicographically ($<$) least paths or short-lex ($<^s$) least paths.

► **Definition 34.** *A homomorphism of finite, pointed, edge-ordered graphs $G \xrightarrow{h} H$ is a homomorphism of pointed graphs that is monotone on neighborhoods; explicitly,*

- (i) $u \rightarrow v$ in G implies $h(u) \rightarrow h(v)$ in H ,
- (ii) v_G is the unique vertex such that $h(v_G) = v_H$, where v_G and v_H are the distinguished points of G and H respectively, and
- (iii) $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ implies $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$.

In addition, a lex-homomorphism must preserve least paths:

- (iv) $h(\min(u \rightsquigarrow v)) = \min(h(u) \rightsquigarrow h(v))$,

and a short-lex homomorphism must preserve least shortest paths:

- (v) $h(\min^s(u \rightsquigarrow v)) = \min^s(h(u) \rightsquigarrow h(v))$.

► **Lemma 35.** *If $h : G \rightarrow H$ is a homomorphism of edge-ordered graphs then h preserves $\cdot < \cdot$ as well as $\cdot <^s \cdot$.*

■ **Table 1** Categories of edge-ordered graphs and their morphisms.

Category	Objects	Morphisms
$\mathbf{FinGraph}_x^<$	finite, pointed, connected edge-ordered graphs	pointed, edge-ordered graph homomorphisms
$\mathbf{FinGraph}_x^l$	finite, pointed, connected edge-ordered graphs	lex-homomorphisms (<i>pointed, edge-ordered graph homomorphisms that preserve least paths</i>)
$\mathbf{FinGraph}_x^s$	finite, pointed, connected edge-ordered graphs	short-lex homomorphisms (<i>pointed, edge-ordered graph homomorphisms that preserve least shortest paths</i>)
$\mathbf{LexGraph}$	lex-graphs (<i>finite, pointed, connected, and edge-ordered by definition</i>)	lex-homomorphisms
$\mathbf{FinArb}_x^<$	finite, pointed, edge-ordered arborescences	pointed, edge-ordered graph homomorphisms
$\mathbf{TLexGraph}$	transitive lex-graphs	lex-homomorphisms

An important special case of edge-ordered graphs are those where the edge order agrees with the path order in each neighborhood.

► **Definition 36.** A lex-graph is a finite, pointed, connected, edge-ordered graph such that the edge order is compatible with the lexicographic path order; explicitly, it is a finite directed graph equipped with

- (i) a distinguished vertex v_0 , and
- (ii) a linear order \triangleleft on each neighbourhood $N(u)$ such that for every $v_1, v_2 \in N(u)$, $\min(u \rightsquigarrow v_1) \prec \min(u \rightsquigarrow v_2) \iff u \rightarrow v_1 \triangleleft u \rightarrow v_2$.

► **Remark 37.** We might be tempted to define, analogously, a *short-lex graph* by demanding that $v_1, v_2 \in N(u)$, $\min^s(u \rightsquigarrow v_1) \prec^s \min^s(u \rightsquigarrow v_2) \iff u \rightarrow v_1 \triangleleft u \rightarrow v_2$. However notice that this condition simply holds automatically for any edge-ordered graph: if there is an edge $u \rightarrow v$, that edge is the lexicographically least shortest path.

Finally, we isolate two extremal special cases of lex-graphs, one with very few edges, and one with very many.

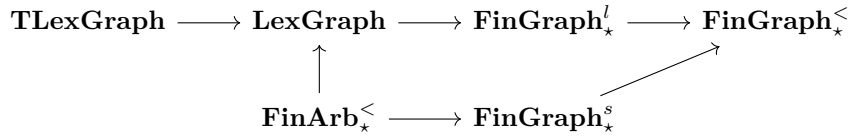
► **Definition 38.** An arborescence is a pointed directed graph $G = (V, \rightarrow, v_0)$ such that for every vertex u , there is a unique path $v_0 \rightsquigarrow u$ in G .

► **Remark 39.** If (V, \rightarrow, v_0) is an arborescence, its underlying undirected graph (V, E) is connected and acyclic (where $E(u, v)$ iff $(u \rightarrow v)$ or $(v \rightarrow u)$), and every edge $u \rightarrow v$ is oriented away from v_0 , meaning that the distance from v_0 to u is less than the distance from v_0 to v . If G is a finite edge-ordered graph that is also an arborescence, then it is already lex-graph, since the only path between u and $v \in N(u)$ is the edge $u \rightarrow v$. Moreover, if S and T are arborescences and if $h : S \rightarrow T$ is a pointed, edge-ordered graph homomorphism, then it is already both a lex-homomorphism and a short-lex homomorphism.

► **Definition 40.** A graph is transitive if its edge relation is; i.e., if $u \rightarrow v$ and $v \rightarrow w$ implies $u \rightarrow w$. A transitive lex-graph is a lex-graph with a transitive edge relation.

It follows readily that this zoo of graph variations each form a category, summarized in Table 1. Diagrammatically, we show the relationships between these categories in Figure 3, where each arrow indicates inclusion as a subcategory.

There is one additional category that will be introduced in Section 8, and that is \mathbf{TArb} (Definition 51) whose objects are transitive closures of arborescences with a particular edge order depending on the underlying arborescence. Curiously, their morphisms preserve *longest* paths instead of shortest paths.



■ **Figure 3** Categories of edge-ordered graphs and their relationships.

5 Least path trees

Given a finite, pointed, connected, edge-ordered graph, we can delete all edges which are not on some lexicographically least path starting from the distinguished vertex v_0 . We get a finite, edge-ordered arborescence. In this section, we show that this construction (denoted Θ) is cofree, indeed coreflective: it is right adjoint to the inclusion functor I .

► **Definition 41** (The functor Θ). *Given a finite, pointed, connected, edge-ordered graph G , with distinguished vertex v_0 , the graph $\Theta(G)$ is defined as follows:*

- *the vertices of $\Theta(G)$ are the vertices of G , and*
- *any edge $u \rightarrow v$ appears in $\Theta(G)$ iff $u \rightarrow v$ is contained in $\min(v_0 \rightsquigarrow v)$.*
- *Order co-initial edges $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $\Theta(G)$ iff the same relation holds in G .*

For a lex-homomorphism $h : G \rightarrow H$ define $\Theta(h) : \Theta(G) \rightarrow \Theta(H)$ by $\Theta(h)(v) = h(v)$ for any vertex $v \in \Theta(G)$.

The proof that Θ is a well-defined functor into the indicated category is in the appendix (Lemma 61). Next we define I , which is simply inclusion of categories.

► **Definition 42** (The functor I). *Given a finite, edge-ordered arborescence T , the object $I(T)$ is simply identified with T . Given a pointed, edge-ordered homomorphism $h : S \rightarrow T$, the homomorphism $I(h) : I(S) \rightarrow I(T)$ is simply identified with h .*

Since every finite, edge-ordered arborescence is a finite, pointed, connected, edge-ordered graph, and since morphisms in $\mathbf{FinArb}_*^<$ are defined to be morphisms of pointed, edge-ordered graphs, I is well-defined.

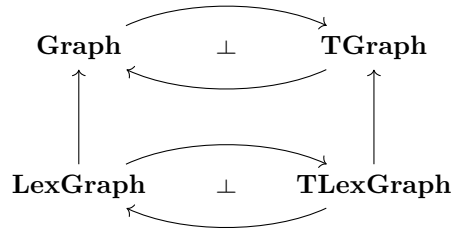
► **Theorem 43.** *There is an adjunction* $\mathbf{FinGraph}_*^l \begin{array}{c} \xrightarrow{\Theta} \\ \top \\ \xleftarrow{I} \end{array} \mathbf{FinArb}_*^<$.

Something for free

Since the image of I is always a lex-graph, and since $\mathbf{LexGraph}$ is a full subcategory of $\mathbf{FinGraph}_*^l$, we actually get the following adjunction for free:

$$\mathbf{LexGraph} \begin{array}{c} \xrightarrow{\Theta'} \\ \top \\ \xleftarrow{I'} \end{array} \mathbf{FinArb}_*^<$$

where I' is the functor I but with target category $\mathbf{LexGraph}$, and Θ' is Θ restricted to $\mathbf{LexGraph}$. The proof is in the appendix (Lemma 62).



■ **Figure 4** The lexicographic-transitive closure of lex-graphs specializes the transitive closure of finite graphs.

6 Least shortest-path trees

Our objective in this section is to establish a shortest-paths analogue of the previous section, i.e., to define a lexicographically least *shortest-path* tree functor $\mathbf{FinGraph}_*^s \xrightarrow{S} \mathbf{FinArb}_*^{<}$ and an inclusion functor $\mathbf{FinArb}_*^{<} \xrightarrow{I} \mathbf{FinGraph}_*^s$. (It is not, of course, the same I as in the previous section, the source category being different.)

► **Definition 44** (The functor S). *Given a finite, pointed, connected, edge-ordered graph G , with distinguished vertex v_0 , the graph $S(G)$ is defined as follows:*

- *the vertices of $S(G)$ are identified with the vertices of G , and*
- *any edge $u \rightarrow v$ appears in $S(G)$ iff $u \rightarrow v$ is contained in $\min^s(v_0 \rightsquigarrow v)$.*
- *Order co-initial edges $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $S(G)$ iff the same relation holds in G .*

If $h : G \rightarrow H$ is a homomorphism, define $S(h) : S(G) \rightarrow S(H)$ by $S(h)(v) = h(v)$, for any vertex $v \in S(G)$.

To show that S is functorial, it suffices to check that $S(G)$ is always an arborescence. We prove this in the appendix (Lemma 63).

► **Theorem 45.** *There is an adjunction* $\mathbf{FinGraph}_*^s \begin{array}{c} \xrightarrow{S} \\ \top \\ \xleftarrow{I} \end{array} \mathbf{FinArb}_*^{<}$.

7 Transitive closure of lex-graphs

There is a well-known adjunction between the category of graphs and the category \mathbf{TGraph} of transitive graphs, where the functor in one direction transitively closes the edge relation, and in the other direction is simply inclusion [11]. Here, we refine this to an adjunction between $\mathbf{LexGraph}$ and $\mathbf{TLexGraph}$. Indeed, the usual transitive closure appears when forgetting the order as in Figure 4 (where the functors $(\mathbf{T})\mathbf{LexGraph} \rightarrow (\mathbf{T})\mathbf{Graph}$ simply forget the edge order).

While it is immediately clear that there is a forgetful (inclusion) functor $\mathbf{TLexGraph} \xrightarrow{U} \mathbf{LexGraph}$, we must construct the free functor in the other direction.

► **Definition 46** (The functor F). *Given a lex-graph G , define the transitive lex-graph $F(G)$ as follows:*

- *the vertices of $F(G)$ are the vertices of G ,*
- *the distinguished point of $F(G)$ is the distinguished point of G ,*
- *The edge relation of $F(G)$ is the transitive closure of the edge relation of G , and finally*

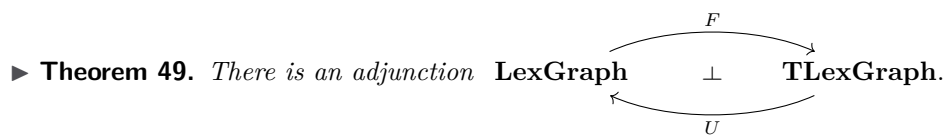
- for any vertices v_1, v_2 in a common neighborhood $N(u)$, let $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $F(G)$ iff $\min(u \rightsquigarrow v_1) \prec \min(u \rightsquigarrow v_2)$ in G .

On morphisms, given $G \xrightarrow{h} H$ we define $F(G) \xrightarrow{F(h)} F(H)$ by $F(h)(v) = h(v)$.

We must show that F is a well-defined functor, but before doing so, we state several important relationships between the lex-graph G and the edge-ordered graph $F(G)$:

- **Lemma 47.** For any lex-graph G ,
 - The neighborhood order of $F(G)$ extends that of G
 - The lexicographic path order of $F(G)$ extends that of G .
 - Least paths in G and $F(G)$ coincide.

- **Lemma 48.** The map $\mathbf{LexGraph} \xrightarrow{F} \mathbf{TLexGraph}$ is well-defined and functorial.



► **Remark 50.** In the title of this paper, we promise universal constructions – and while such do arise from adjunctions (from the unit and counit), it seems fitting that we make this explicit at least once. As a consequence of Theorem 49, for any lex-graph G in with lexicographic-transitive closure $U(F(G))$, given any other transitive lex-graph $U(G')$ and $G \xrightarrow{h} U(G')$ there is a *unique* homomorphism of transitive lex-graphs $F(G) \xrightarrow{\hat{h}} G'$ such that the diagram below commutes in $\mathbf{LexGraph}$:

$$\begin{array}{ccc} G & \xrightarrow{\quad} & U(F(G)) \\ & \searrow h & \downarrow U(\hat{h}) \\ & & U(G') \end{array}$$

The reader is invited to extract similar universal mapping properties from other adjunctions in this paper.

8 Transitive closure of least shortest path trees

We now establish the final adjunction of our paper, and the second adjunction needed to characterize breadth-first traversals. Analogously to the depth-first case, this adjunction relates $\mathbf{FinArb}_*^{\prec}$ and a category of transitively closed graphs:

► **Definition 51** (The category \mathbf{TArb}). A finite, pointed, connected, edge-ordered graph G is an object of \mathbf{TArb} iff there exists a finite, pointed, connected, edge-ordered arborescence T and an identification of the vertices of G with the vertices of T such that

- the distinguished points of each are identical,
- the edge relation of G is the transitive closure of the edge relation of T , and
- $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in G iff $v_0 \rightsquigarrow v_1 \prec^s v_0 \rightsquigarrow v_2$ in T , where $v_0 \rightsquigarrow v$ is the unique path from v_0 to v in T .

A map $G_1 \xrightarrow{h} G_2$ is a morphism in \mathbf{TArb} if it is a homomorphism of edge-ordered graphs ((i)–(iii) of Definition 34) and in addition it preserves longest paths.

This last property only makes sense if longest paths are unique. Luckily, longest paths in the transitive closure of an arborescence are exactly the edges of the original tree.

17:12 Graph Traversals as Universal Constructions

► **Lemma 52.** *If T is an arborescence (not necessarily edge-ordered) with root v_0 , and G is its transitive closure, then an edge $u \rightarrow v$ of G is an edge of T iff $u \rightarrow v$ appears on a longest path $v_0 \rightsquigarrow v$ in G . Moreover, the longest path $u \rightsquigarrow v$ in G is unique.*

► **Remark 53.** As a consequence of Lemma 52, for any graph $G \in \mathbf{TArb}$, $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in G iff $v_0 \rightsquigarrow v_1 \prec^s v_0 \rightsquigarrow v_2$ in G , where $v_0 \rightsquigarrow v$ is the unique longest path from v_0 to v in G .

While the presentation of depth- and breadth-first traversals has thus far been similar even at a rather small scale, this adjunction diverges from the previous one in several ways: it does not decompose into an inclusion and a functor which lifts the ordinary transitive closure, and the curious preservation of longest paths by morphisms is not suggested by the pre-categorical treatment of breadth-first traversals. This category arises somewhat mysteriously, and we have no justification for introducing it other than it works.

We now define the two functors of the adjunction. The proofs that they are functorial are Lemmas 64 and 65 in the appendix.

► **Definition 54.** *Define the functor $\mathbf{FinArb}_*^{\leq} \xrightarrow{\Gamma} \mathbf{TArb}$ by $\Gamma(V, v_0, \rightarrow) = (V, v_0, \xrightarrow{\text{trans}})$, and for any vertices (u, v_1, v_2) such that $u \rightarrow v_1$ and $u \rightarrow v_2$ in $\Gamma(T)$, define $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ iff $v_0 \rightsquigarrow v_1 \prec^s v_0 \rightsquigarrow v_2$ in T . For morphisms $T \xrightarrow{h} T'$, define $\Gamma(h) : \Gamma(T) \rightarrow \Gamma(T')$ by $\Gamma(h)(v) = h(v)$.*

► **Definition 55.** *Define the functor $\mathbf{TArb} \xrightarrow{L} \mathbf{FinArb}_*^{\leq}$ by identifying the set of vertices of $L(G)$ with the vertices of G , and including an edge $u \rightarrow v$ in $L(G)$ iff it lies on the longest path $v_0 \rightsquigarrow v$ in G . The neighborhood order in $L(G)$ is inherited from G , and for morphisms $G \xrightarrow{h} H$, define $L(h) : L(G) \rightarrow L(H)$ by $L(h)(v) = h(v)$.*

► **Theorem 56.** *There is an adjunction $\mathbf{FinArb}_*^{\leq} \begin{array}{c} \xrightarrow{\Gamma} \\ \perp \\ \xleftarrow{L} \end{array} \mathbf{TArb}$.*

9 Putting it all together

Combining the adjunctions from Sections 5 and 7, we get a chain of adjunctions

$$\mathbf{FinGraph}_*^{\leq} \begin{array}{c} \xrightarrow{\Theta} \\ \top \\ \xleftarrow{I} \end{array} \mathbf{FinArb}_*^{\leq} \begin{array}{c} \xrightarrow{I'} \\ \perp \\ \xleftarrow{\Theta'} \end{array} \mathbf{LexGraph} \begin{array}{c} \xrightarrow{F} \\ \perp \\ \xleftarrow{U} \end{array} \mathbf{TLexGraph}$$

Since adjunctions of the same handedness compose, we can rewrite this as

$$\mathbf{FinGraph}_*^{\leq} \begin{array}{c} \xrightarrow{\Theta} \\ \top \\ \xleftarrow{I} \end{array} \mathbf{FinArb}_*^{\leq} \begin{array}{c} \xrightarrow{I' \circ F} \\ \top \\ \xleftarrow{\Theta' \circ U} \end{array} \mathbf{TLexGraph}$$

Similarly, by combining the adjunctions of Sections 6 and 8 we get

$$\mathbf{FinGraph}_*^s \begin{array}{c} \xrightarrow{S} \\ \top \\ \xleftarrow{I} \end{array} \mathbf{FinArb}_*^{\leq} \begin{array}{c} \xrightarrow{\Gamma} \\ \top \\ \xleftarrow{L} \end{array} \mathbf{TArb}$$

Suppose we fix a finite, pointed, edge-ordered graph, locate it in either $\mathbf{FinGraph}_*^l$ or $\mathbf{FinGraph}_*^s$, then apply the appropriate least-path tree and the transitive closure functors. Then the edge ordering of the distinguished vertex in the result is the lexicographic depth-first or breadth-first traversal respectively.

► **Lemma 57.** *Given any $G \in \mathbf{FinGraph}_*^l$, let $T = (F \circ I' \circ \Theta)(G)$. Then the neighborhood ordering \triangleleft on the distinguished point in T is the lexicographic depth-first traversal of G .*

► **Lemma 58.** *Given any $G \in \mathbf{FinGraph}_*^s$, let $T = (\Gamma \circ S)(G)$. Then the neighborhood ordering \triangleleft on the distinguished point in T is the lexicographic breadth-first traversal of G .*

Finally, we show that these traversals can be extracted as vertex orders rather than edge orders, by defining a functor that takes an edge-ordered graph into the ordered neighborhood of its distinguished point. Let $\mathbf{FinLoSet}$ denote the category of finite linearly ordered sets with monotone functions.

► **Lemma 59.** *There is a functor $E : \mathbf{FinGraph}_*^< \rightarrow \mathbf{FinLoSet}$ that linearly orders the vertices in a given graph according to the ordering on $N(v_0)$.*

Since there are inclusions $\mathbf{TLexGraph} \rightarrow \mathbf{FinGraph}_*^<$ and $\mathbf{TArb} \rightarrow \mathbf{FinGraph}_*^<$, we get

► **Corollary 60.** *There are functors $\mathbf{FinGraph}_*^l \rightarrow \mathbf{FinLoSet}$ and $\mathbf{FinGraph}_*^s \rightarrow \mathbf{FinLoSet}$ which compute the lexicographic depth-first and breadth-first traversals respectively.*

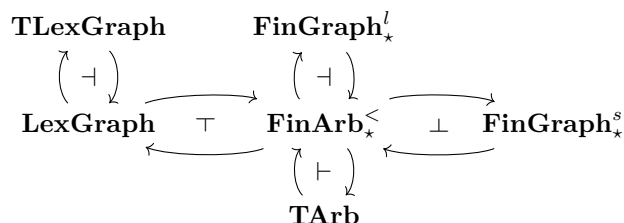
Note that parts of this final step $\mathbf{FinGraph}_*^< \rightarrow \mathbf{FinLoSet}$ can also be made universal, but since we still have to accept the presence of inclusion functors without adjoints, we only sketch this construction: The functor E from Lemma 59 restricts to a functor from the category of *transitive* finite, pointed, edge-ordered graphs and their homomorphisms, and into the category of finite *nonempty* linearly ordered sets with monotone functions *which additionally preserve the least element*. This restricted functor has a left adjoint given by mapping a finite linearly ordered set $(V, <)$ to the graph with vertices V , distinguished vertex the least element v_0 of V (guaranteed to exist when V is finite), and edge relation given by $v_0 \rightarrow v$ for all $v \in V$ with $v \neq v_0$. This specializes to neither $\mathbf{TLexGraph}$ nor \mathbf{TArb} , however, since the counit of this adjunction need not preserve either least or longest paths.

10 Discussion and open questions

We have described a construction of depth-first traversals from a category of finite edge-ordered graphs whose morphisms preserve least paths as a series of universal constructions, and in analogous fashion, a construction of breadth-first traversals from a category of finite edge-ordered graphs whose morphisms preserve least shortest paths. A diagram summarizing the various adjunctions involved is shown in Figure 5. The prevalence of coreflections in our work is nicely aligned with previous work on reachability in coalgebras [2, 14], where well-pointed and reachable coalgebras respectively turn out to form coreflective subcategories.

This begs the question of whether there is way of generalizing these two constructions using a general notion of extremal path. Such a method is suggested by Delatorre and Kruskal [7] using the machinery of a closed semiring system, which provides a general setting for several lexicographic algebraic path problems; not only breadth-first and depth-first search, but *lexicographic topological search* as well.

The deeper question raised by our work is how, if at all, the categorification of a problem informs the algorithms which solve it. For example, it seems natural that problems which can be expressed by a *single* (free or cofree) universal construction would be amenable to a



■ **Figure 5** Categories of edge-ordered graphs and adjunctions between them.

solution by a greedy algorithm. Intuitively, if an object is free, then it can be built up *over here* without affecting what happens *over there*, and can thus be constructed by making local choices – exactly what a greedy algorithm does.

What if a problem can be expressed by a composition of two adjunctions of opposite handedness, as we show here? Is there an algorithmic strategy for such problems? More generally, is there a robust hierarchy of problems on graphs which classifies them by the number of free-cofree alternations, and can we unite problems in the same level of the hierarchy with a common algorithmic template?

Speculatively, we envision a world in which properties of algorithms could be inferred from the appropriate categorical formulation of the problems they solve. In such a world, a statement like *problem X can be solved using two sequential applications of depth-first search, but no fewer* would have a precise meaning. The semantics functor that transforms an algorithm into a problem that it solves could be decomposed in several meaningful ways, which would give different pieces of information about the algorithm. (Perhaps, for example, one decomposition would give upper bounds on the sequential complexity of solving the problem, and another on the parallel complexity.)

This is a long-term goal. In the medium term, there are still many compelling questions we might hope to answer. For example, the common sequential algorithm implementing depth-first search uses stacks, and the analogous algorithm for breadth-first search uses queues. Can we somehow infer “stacks” and “queues” from a common categorical formulation of depth-first and breadth-first traversals? Can we recover the parallel complexities of these problems proven in [6]?

Positive answers to these or similar questions would be strong evidence that the “categorical structure theory of algorithms” mentioned in the introduction is actually a viable program; that it can not only describe problems, but suggest ways to solve them – in the elegant characterization of [1], to *lead* rather than to *follow*.

References

- 1 S. Abramsky. Whither semantics? *Theoretical Computer Science*, 807:3–14, 2020.
- 2 J. Adámek, S. Milius, L. S. Moss, and L. Sousa. Well-pointed coalgebras. *Logical Methods in Computer Science*, 9(3), 2013.
- 3 E. Allender, A. Chauhan, and S. Datta. Depth-first search in directed graphs, revisited. *Electronic colloquium on computational complexity*, 20(74), 2020.
- 4 S. Bhaskar and R. Kaarsgaard. Graph traversals as universal constructions, 2021. [arXiv: 2104.14877](https://arxiv.org/abs/2104.14877).
- 5 D. Corneil and R. Krueger. A unified view of graph searching. *SIAM J. Discrete Math.*, 22:1259–1276, January 2008.

- 6 P. Delatorre and C. P. Kruskal. Polynomially improved efficiency for fast parallel single-source lexicographic depth-first search, breadth-first search, and topological-first search. *Theory of Computing Systems*, 34:275–298, 2001.
- 7 P. Delatorre and C.P. Kruskal. Fast parallel algorithms for all-sources lexicographic search and path-algebra problems. *Journal of Algorithms*, 19(1):1–24, 1995.
- 8 L. Dixon and A. Kissinger. Open-graphs and monoidal theories. *Mathematical Structures in Computer Science*, 23(2):308–359, 2013.
- 9 J. Kock. Graphs, hypergraphs, and properads. *Collectanea mathematica*, 67(2):155–190, 2016.
- 10 S. Lack and P. Sobociński. Adhesive and quasiadhesive categories. *RAIRO-Theoretical Informatics and Applications*, 39(3):511–545, 2005.
- 11 J. Master. The open algebraic path problem, 2020. [arXiv:2005.06682](https://arxiv.org/abs/2005.06682).
- 12 J. Rathke, P. Sobociński, and O. Stephens. Compositional reachability in petri nets. In J. Ouaknine, I. Potapov, and J. Worrell, editors, *Reachability Problems*, pages 230–243. Springer, 2014.
- 13 D. J. Rose and R. E. Tarjan. Algorithmic aspects of vertex elimination. In *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing (STOC '75)*, pages 245–254. ACM, 1975.
- 14 T. Wißmann, S. Milius, S. Katsumata, and J. Dubut. A coalgebraic view on reachability. *Commentationes Mathematicae Universitatis Carolinae*, 60(4):605–638, 2019.

A Appendix: Proofs from Sections 4 through 9

Proof of Lemma 35. It suffices to show that h preserves $\cdot \prec \cdot$; the second statement follows from the observation that graph homomorphisms preserve path length, i.e., $|\pi| = |h(\pi)|$ for every path π .

Suppose that π and σ are co-initial paths in G such that $\pi \prec \sigma$. If $\pi \sqsubset \sigma$, then $h(\pi) \sqsubset h(\sigma)$ and we're done. Otherwise, let $u \rightarrow v_1$ and $u \rightarrow v_2$ be the first edges in π and σ respectively following their longest common prefix ζ . Since $\pi \prec \sigma$, $u \rightarrow v_1 \triangleleft u \rightarrow v_2$, so $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$.

Notice that $h(\zeta)$ is a common prefix of $h(\pi)$ and $h(\sigma)$, and since $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$, it must be the longest one. Moreover, since $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$, $h(\pi) \prec h(\sigma)$, which is what we wanted to show. \blacktriangleleft

► **Lemma 61.** *The functor Θ is well-defined.*

Proof. We first have to check that $\Theta(G)$ is a finite, edge-ordered arborescence. It is trivially finite and edge-ordered. Notice that $\Theta(G)$ is connected: for every vertex u , every edge on the path $\min(v_0 \rightsquigarrow u)$ is contained in $\Theta(G)$. On the other hand, the in-degree of each vertex is at most 1: there cannot be two distinct edges $u \rightarrow v$ and $u' \rightarrow v$ in $\min(v_0 \rightsquigarrow v)$.

Next we have to check that for any homomorphism h , $\Theta(h)$ is in fact a homomorphism of pointed, edge-ordered graphs. First notice that $\Theta(h)$ actually maps into $\Theta(H)$: if $u \rightarrow v$ is included in $\Theta(G)$, then it must be contained in $\min(v_0 \rightsquigarrow v)$ in G , whose h -image is $\min(h(v_0) \rightsquigarrow h(v))$ by property (iv) of Definition 34; therefore, $h(u \rightarrow v)$ is contained in a least path and so included in $\Theta(H)$.

Moreover, $\Theta(h)$ clearly fixes the distinguished vertex, and visibly inherits monotonicity on co-initial edges from h . Hence, $\Theta(h)$ is a pointed, edge-ordered graph homomorphism. \blacktriangleleft

Proof of Theorem 43. First, given $I(T) \xrightarrow{h^\uparrow} G$ in $\mathbf{FinGraph}_*^l$, we describe how to obtain $T \xrightarrow{h_\downarrow} \Theta(G)$ in $\mathbf{FinArb}_*^<$. Define $h_\downarrow(v) = h^\uparrow(v)$, for $v \in T$ (which, remember is the same as $I(T)$). To check that h_\downarrow is well-defined, we must show that if $u \rightarrow v$ is an edge of T , then

17:16 Graph Traversals as Universal Constructions

$h^\uparrow(u \rightarrow v)$ is contained in $\Theta(G)$. But since T is an arborescence, $u \rightarrow v$ is trivially contained in $\min(v_0 \rightsquigarrow v)$, so $h^\uparrow(u \rightarrow v)$ is contained in $\min(h(v_0) \rightsquigarrow h(v))$, and hence included in $\Theta(G)$.

Moreover, the fact that h_\downarrow maps edges to edges, preserves the distinguished point, and is monotone on co-initial edges, is immediately inherited from h^\uparrow .

Next, given $T \xrightarrow{h^\downarrow} \Theta(G)$ in $\mathbf{FinArb}_*^<$, define $I(T) \xrightarrow{h^\uparrow} G$ in $\mathbf{FinGraph}_*^l$ by $h^\uparrow(v) = h_\downarrow(v)$, for $v \in I(T)$. In this case, we do not need to check that h^\uparrow is well-defined, and the properties of mapping edges to edges, preserving the distinguished point, and monotonicity on neighborhoods, are inherited immediately from h_\downarrow . The fact that h^\uparrow maps least paths to least paths is easily justified by observing that h^\uparrow maps into $\Theta(G)$, the tree of least paths in G .

To check that this correspondence is bijective, notice that starting from either h^\uparrow or h_\downarrow , passing to the other one, and passing back again, gives us the same morphism we started with. Naturality follows straightforwardly by observing that both functors are the identity on morphisms, so naturality squares trivially commute. \blacktriangleleft

► **Lemma 62.** *There is an adjunction*

$$\mathbf{LexGraph} \quad \begin{array}{c} \xrightarrow{\Theta'} \\ \top \\ \xleftarrow{I'} \end{array} \quad \mathbf{FinArb}_*^<$$

Proof. The adjunction from Theorem 43 factors as

$$\begin{array}{ccc} \mathbf{LexGraph} & \xrightarrow{J} & \mathbf{FinGraph}_*^l \\ & \searrow^{I'} & \uparrow \left(\begin{array}{c} \dashv \\ \ominus \end{array} \right) \\ & & \mathbf{FinArb}_*^< \end{array}$$

where J is the fully faithful identity-on-objects functor witnessing the inclusion of $\mathbf{LexGraph}$ in $\mathbf{FinGraph}_*^l$, and $\Theta' = \Theta \circ J$. The natural isomorphism

$$\begin{aligned} \mathbf{LexGraph}(I'(G), H) &\cong \mathbf{FinGraph}_*^l(J(I'(G)), J(H)) \\ &= \mathbf{FinGraph}_*^l(I(G), J(H)) \\ &\cong \mathbf{FinArb}_*^<(G, \Theta(J(H))) \\ &= \mathbf{FinArb}_*^<(G, \Theta'(H)) \end{aligned}$$

establishes this adjunction. \blacktriangleleft

► **Lemma 63.** *For any finite, pointed, connected, edge-ordered graph G , $S(G)$ is an arborescence.*

Proof. To verify that $S(G)$ is a well-defined arborescence, it suffices to observe that $S(G)$ is connected (as it preserves least shortest paths), and that there is a unique path $v_0 \rightsquigarrow v$: two distinct edges $u \rightarrow v$ and $u' \rightarrow v$ cannot both occur in $\min^s(v_0 \rightsquigarrow v)$.

To check that $S(G \xrightarrow{h} H)$ is well defined, we have to verify for $u \rightarrow v \in S(G)$, $h(u) \rightarrow h(v) \in S(H)$. But such morphisms h preserve least shortest paths by Definition 34 (v). \blacktriangleleft

Proof of Theorem 45. The proof is obtained by literally copying the proof of Theorem 43 and replacing $\mathbf{FinGraph}_*^l$ by $\mathbf{FinGraph}_*^s$, Θ by S , and \min by \min^s throughout.

Given $I(T) \xrightarrow{h^\uparrow} G$ in $\mathbf{FinGraph}_*^s$, we describe how to obtain $T \xrightarrow{h_\downarrow} T(G)$ in $\mathbf{FinArb}_*^<$. Define $h_\downarrow(v) = h^\uparrow(v)$, for $v \in T$ (which we identified with $I(T)$). To check that h_\downarrow is well-defined, we must show that if $u \rightarrow v$ is an edge of T , then $h^\uparrow(u \rightarrow v)$ is contained in $S(G)$. But since T is an arborescence, $u \rightarrow v$ is trivially contained in $\min^s(v_0 \rightsquigarrow v)$, so $h^\uparrow(u \rightarrow v)$ is contained in $\min^s(h(v_0) \rightsquigarrow h(v))$, and hence included in $S(G)$.

Moreover, the fact that h_\downarrow maps edges to edges, preserves the distinguished point, and is monotone on co-initial edges, is immediately inherited from h^\uparrow .

Next, given $T \xrightarrow{h_\downarrow} S(G)$ in $\mathbf{FinArb}_*^<$, define $I(T) \xrightarrow{h^\uparrow} G$ in $\mathbf{FinGraph}_*^s$ by $h^\uparrow(v) = h_\downarrow(v)$, for $v \in I(T)$ (which we identified with T). In this case, we do not need to check that h^\uparrow is well-defined, and the properties of mapping edges to edges, preserving the distinguished point, and monotonicity on neighborhoods, are inherited immediately from h_\downarrow . The fact that h^\uparrow maps least shortest paths to least shortest paths is easily justified by observing that h^\uparrow maps into $S(G)$, the tree of least shortest paths in G .

To check that this correspondence is bijective, notice that starting from either h^\uparrow or h_\downarrow , passing to the other one, and passing back again, gives us the same morphism we started with. Naturality follows straightforwardly by observing that both functors are the identity on morphisms, so naturality squares trivially commute. \blacktriangleleft

Proof of Lemma 47. (i): If there are edges $u \rightarrow v_1$ and $u \rightarrow v_2$ of G , then $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in G iff $\min(u \rightsquigarrow v_1) \prec \min(u \rightsquigarrow v_2)$ in G (since G is a lex-graph) iff $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $F(G)$ (by definition of F).

(ii): Suppose σ and π are co-initial paths from G . We may assume that σ and π share no nontrivial prefix; then either σ or π is empty (and the claim is trivial), or σ and π differ on their first edge, and the claim follows from (i).

(iii): Work in $F(G)$. It suffices to show that least paths in $F(G)$ consist of only edges in G ; by the above remark, if two paths consist of G -edges, it does not matter whether we compare them in G or in $F(G)$.

Towards which, it suffices to show that every edge $u \rightarrow v$ not in G is greater than the least path $\min_G(u \rightsquigarrow v)$ between u and v in G . Then any path with non-edges in G can be lessened; in particular least paths in $F(G)$ cannot contain any non-edges of G .

Let $v_1 \neq v$ be the second vertex in $\min_G(u \rightsquigarrow v)$. Since least paths are closed under taking prefixes, $u \rightarrow v_1 = \min_G(u \rightsquigarrow v_1)$, in both G and $F(G)$. Since $\min_G(u \rightsquigarrow v_1) \prec \min_G(u \rightsquigarrow v)$, $u \rightarrow v_1 \triangleleft u \rightarrow v$ in $F(G)$. But then $\min_G(u \rightsquigarrow v) \prec u \rightarrow v$, which is what we wanted to show. \blacktriangleleft

Proof of Lemma 48. Given any lex-graph G , $F(G)$ is clearly a transitive edge-ordered graph, but we must check that it satisfies the lex-graph property (Definition 36-(ii)). But for any v_1 and v_2 in the neighborhood of a common u in $F(G)$, $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $F(G)$ iff $\min(u \rightsquigarrow v_1) \prec \min(u \rightsquigarrow v_2)$ in G (by definition of F) iff $\min(u \rightsquigarrow v_1) \prec \min(u \rightsquigarrow v_2)$ in $F(G)$ (by Lemma 47 plus the preceding remark).

By definition, F immediately preserves identities and compositions of homomorphisms, and it remains to check that for any lex-homomorphism $G \xrightarrow{h} H$ of lex-graphs, $F(h)$ is a lex-homomorphism $F(G) \rightarrow F(H)$. We check each of the conditions in Definition 34:

- (i) If $u \rightarrow v$ in $F(G)$, then there is a path $u \rightsquigarrow v$ in G , so there is a path $h(u) \rightsquigarrow h(v)$ in H , and hence an edge $h(u) \rightarrow h(v)$ in $F(H)$.
- (ii) Since h preserves the distinguished point, so does $F(h)$.
- (iii) If $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in G , $\min(u \rightsquigarrow v_1) \prec \min(u \rightsquigarrow v_2)$, as G is a lex-graph. Since h is a lex-homomorphism, $\min_H(h(u) \rightsquigarrow h(v_i)) = h(\min_G(u \rightsquigarrow v_i))$, so $\min(h(u) \rightsquigarrow h(v_1)) \prec \min(h(u) \rightsquigarrow h(v_2))$ in H . By definition of F , $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$ in $F(H)$.

17:18 Graph Traversals as Universal Constructions

- (iv) As the least path $\min(u \rightsquigarrow v)$ in $F(G)$ is also least in G , as shown above, and as h preserves least paths, $h(\min(u \rightsquigarrow v))$ is $\min(h(u) \rightsquigarrow h(v))$ in H , and this in turn is also the least path in $F(H)$. ◀

Proof of Theorem 49. Let $F(G) \xrightarrow{h^\uparrow} H$ be a homomorphism of **TLexGraph**. Since $F(G)$ has the same vertices as G , we may define $G \xrightarrow{h_\downarrow} U(H)$ by $h_\downarrow(v) = h^\uparrow(v)$ (since H and $U(H)$ are exactly identical).

We check that h_\downarrow is a lex-homomorphism by checking Definition 34 (i)-(iv). We write, e.g., $h(v)$ to refer unambiguously to the vertex $h^\uparrow(v) = h_\downarrow(v)$.

- (i) If $u \rightarrow v$ is an edge of G , it is an edge of $F(G)$, so $h(u) \rightarrow h(v)$ is an edge of H , and hence an edge of $U(H)$.
- (ii) h_\downarrow directly inherits preservation of the distinguished point from h^\uparrow
- (iii) If $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in G , then $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $F(G)$ (Lemma 47), so $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$ in H (monotonicity of h^\uparrow), and hence the same holds in $U(H)$.
- (iv) If π is the least path $u \rightsquigarrow v$ in G , then it's least in $F(G)$ (Lemma 47), so $h(\pi)$ is least in H (since h^\uparrow is a lex-homomorphism), and hence also in $U(H)$.

In the other direction, we suppose that we are given some homomorphism of lex-graphs $G \xrightarrow{h_\downarrow} U(H)$ in **LexGraph**. Define $F(G) \xrightarrow{h^\uparrow} H$ by $h^\uparrow(v) = h_\downarrow(v)$. Again, we check (i)-(iv) of Definition 34.

- (i) If $u \rightarrow v$ is an edge of $F(G)$, then there is a path $u \rightsquigarrow v$ in G , and hence a path $h(u) \rightsquigarrow h(v)$ in $U(H)$, and (since H is transitive), and edge $h(u) \rightarrow h(v)$.
- (ii) h^\uparrow directly inherits preservation of the distinguished point from h_\downarrow
- (iii) If $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $F(G)$, then $\min(u \rightsquigarrow v_1) \prec \min(u \rightsquigarrow v_2)$ in G ; hence (since h_\downarrow is a lex-homomorphism) $\min(h(u) \rightsquigarrow h(v_1)) \prec \min(h(u) \rightsquigarrow h(v_2))$ in $U(H)$, hence in H . Since H is a lex-graph, $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$.
- (iv) If π is the least path $u \rightsquigarrow v$ in $F(G)$, then it's the least path in G (Lemma 47), so $h(\pi)$ is least in $U(H)$ (since h_\downarrow is a lex-homomorphism), and hence in H .

We need only now to check that this correspondence is bijective and natural. Bijectivity follows readily from the fact that we always have $h^\uparrow = h_\downarrow$, so going from $G \xrightarrow{h_\downarrow} U(H)$ to $F(G) \xrightarrow{h^\uparrow} H$ and back has no effect, and similarly in the other direction. Similarly, naturality follows straightforwardly by noting that $U(h)(v) = F(h)(v) = h(v)$ for all h and v , so naturality squares trivially commute. ◀

Proof of Lemma 52. Notice that distances between vertices are never increased in G compared to T , only decreased, meaning that if there is a path $u \rightsquigarrow v$ in G , there is a path $u \rightsquigarrow v$ in T that is no shorter. Therefore, longest paths in G must consist entirely of edges in T , and are therefore unique.

Conversely, if $u \rightarrow v$ is an edge of T , then it appears on the unique path $v_0 \rightsquigarrow v$ in T . As just observed, the longest path $v_0 \rightsquigarrow v$ in G is also a path in T ; hence, it is the unique path $v_0 \rightsquigarrow v$ in T , and thus contains $u \rightarrow v$. ◀

► **Lemma 64.** Γ is a well-defined functor.

Proof. Clearly Γ preserves identities and composition. We must check that for any morphism h of $\mathbf{FinArb}_*^<$, $\Gamma(h)$ satisfies Definition 34 (i)-(iii) and preserves longest paths:

- (i) If $u \rightarrow v$ in $\Gamma(T)$, then $u \rightsquigarrow v$ in T , so $h(u) \rightsquigarrow h(v)$ in T' , so $h(u) \rightarrow h(v)$ in $\Gamma(T')$.
- (ii) Since h maps the distinguished point, and only that point, of T to the distinguished point of T' , Γ does the same from $\Gamma(T)$ to $\Gamma(T')$.

- (iii) If $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $\Gamma(T)$, then $v_0 \rightsquigarrow v_1 \prec^s v_0 \rightsquigarrow v_2$ in T . By Lemma 35, $h(v_0 \rightsquigarrow v_1) \prec^s h(v_0 \rightsquigarrow v_2)$ in T' , and since paths in arborescences are unique, $h(u) \rightsquigarrow h(v_1) \prec^s h(u) \rightsquigarrow h(v_2)$ in T' . Hence $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$ in $\Gamma(T')$.
- Finally, if $u \rightsquigarrow v$ is the longest path in $\Gamma(T)$, then it is a path in T by Lemma 52, and therefore, $h(u \rightsquigarrow v) = h(u) \rightsquigarrow h(v)$ is a path in T' . Again by Lemma 52, $h(u \rightsquigarrow v)$ is the longest path in $\Gamma(T')$. ◀

► **Lemma 65.** L is a well-defined functor $\mathbf{TArb} \rightarrow \mathbf{FinArb}_*^{\leq}$.

Proof. In each graph $G \in \mathbf{TArb}$, the unique longest paths are closed under taking prefixes. Therefore, the union of all least longest paths forms an arborescence.

To check that L is a functor, note that for any morphism $h \in \mathbf{TArb}$, $L(h)$ clearly preserves the distinguished point and is monotone on neighborhoods. We only need to show that if $u \rightarrow v$ is an edge in $L(G)$ and $h : G \rightarrow H$ is a morphism in \mathbf{TArb} , then $h(u) \rightarrow h(v)$ is an edge of $L(H)$. But, this is guaranteed by the fact that h preserves longest paths.

Finally, note that L preserves the identity morphism and respects composition. ◀

Proof of Theorem 56. Fix a pointed, connected, edge-ordered arborescence T and a pointed, connected, transitive, edge-ordered graph G .

Given $\Gamma(T) \xrightarrow{h^\uparrow} G$ in \mathbf{TArb} , we define $T \xrightarrow{h_\downarrow} L(G)$ by $h_\downarrow(v) = h^\uparrow(v)$. This is well-defined, because if $u \rightarrow v$ is an edge in T , then by Lemma 52, it appears on the unique longest path $v_0 \rightsquigarrow v$ in $\Gamma(T)$. Since h^\uparrow preserves least longest paths, the edge $u \rightarrow v$ maps into $L(G)$. Moreover, h_\downarrow preserves the distinguished point and inherits monotonicity in neighborhoods from h^\uparrow , so satisfies the conditions of Definition 34 and is a morphism in \mathbf{FinArb}_*^{\leq} .

In the other direction, given $T \xrightarrow{h_\downarrow} L(G)$ in \mathbf{FinArb}_*^{\leq} , we define $\Gamma(T) \xrightarrow{h^\uparrow} G$ by $h^\uparrow(v) = h_\downarrow(v)$; let us unambiguously write $h(v)$ for brevity. If $u \rightarrow v$ is an edge of $\Gamma(T)$, then there is a path $u \rightsquigarrow v$ in T , hence a path $h(u) \rightsquigarrow h(v)$ in $L(G)$, and therefore an edge $h(u) \rightarrow h(v)$ in G . Moreover, if $u \rightarrow v_1 \triangleleft u \rightarrow v_2$ in $\Gamma(T)$, then $v_0 \rightsquigarrow v_1 \prec^s v_0 \rightsquigarrow v_2$ in T , so $h(u) \rightsquigarrow h(v_1) \prec^s h(u) \rightsquigarrow h(v_2)$ in $L(G)$, by Lemma 35. By the remark succeeding Lemma 52, $h(u) \rightarrow h(v_1) \triangleleft h(u) \rightarrow h(v_2)$ in G . Finally, if $u \rightsquigarrow v$ is the longest path from u to v in $\Gamma(T)$, then each of its edges lies in T by Lemma 52, hence its h -image lies in $L(G)$, which means it is a longest path of G . Therefore, h^\uparrow is a morphism of \mathbf{TArb} .

It remains to show that the maps relating h^\uparrow and h_\downarrow are bijective and natural. As in the proof of Theorem 49, this is immediate from the definition of each map; the only thing to show is that they were well-defined. ◀

Proof of Lemma 57. Fix $u, v \neq v_0$. By definition of F , $v_0 \rightarrow u \triangleleft v_0 \rightarrow v$ in T iff $\min(v_0 \rightsquigarrow u) \prec \min(v_0 \rightsquigarrow v)$ in $(I' \circ \Theta)(G)$. Since I' is an inclusion functor, this is equivalent to $\min(v_0 \rightsquigarrow u) \prec \min(v_0 \rightsquigarrow v)$ in $\Theta(G)$; indeed, the *unique* path $v_0 \rightsquigarrow u$ is less than the *unique* path $v_0 \rightsquigarrow v$ in $\Theta(G)$.

But the unique paths $v_0 \rightsquigarrow u$ and $v_0 \rightsquigarrow v$ in $\Theta(G)$ are exactly $\min(v_0 \rightsquigarrow u)$ and $\min(v_0 \rightsquigarrow v)$ in G respectively; moreover, the relative order on the latter two paths in G is inherited from the relative order on the former two in $\Theta(G)$.

Therefore, $v_0 \rightarrow u \triangleleft v_0 \rightarrow v$ in T iff $\min(v_0 \rightsquigarrow u) \prec \min(v_0 \rightsquigarrow v)$ in G , but this is exactly the relation $<_D$ of Definition 28, which is the lexicographic depth-first traversal of G by Theorem 30. ◀

Proof of 58. Fix $u, v \neq v_0$. By definition of Γ , $v_0 \rightarrow u \triangleleft v_0 \rightarrow v$ in T iff $v_0 \rightsquigarrow u \prec^s v_0 \rightsquigarrow v$ in $S(G)$ (where paths from v_0 are unique). By definition of S , this is equivalent to $\min^s(v_0 \rightsquigarrow u) \prec^s \min^s(v_0 \rightsquigarrow v)$ in G . By Corollary 27, this is equivalent to $u <_B v$. ◀

17:20 Graph Traversals as Universal Constructions

Proof of Lemma 59. For a finite, edge-ordered graph G , we define $E(G)$ to be the order $(\{v_0\} \cup N(v_0), <)$, where for $u, v \neq v_0$, $u < v \iff v_0 \rightarrow u \triangleleft v_0 \rightarrow v$, and for $u \neq v_0$, $v_0 < u$.

On morphisms, given a homomorphism of edge-ordered graphs $G \xrightarrow{h} H$, we define $E(G) \xrightarrow{E(h)} E(H)$ by $E(h)(v) = h(v)$. Notice that E is well-defined, since it maps the distinguished point v_0 of G to the distinguished point w_0 of H , and also maps $N_G(v_0)$ into $N_H(w_0)$. By definition, it is clear that E preserves both identities and compositions, so we have only left to show that $E(h)$ is monotone.

Since h maps only v_0 to w_0 , it suffices to show that if $u, v \in N_G(v_0)$ and $v_0 \rightarrow u \triangleleft v_0 \rightarrow v$ in G , then $w_0 \rightarrow h(u) \triangleleft w_0 \rightarrow h(v)$ in H . But this follows from monotonicity of h (Definition 34-(iii)) ◀

Space-Efficient Fault-Tolerant Diameter Oracles

Davide Bilò  

Department of Humanities and Social Sciences, University of Sassari, Italy

Sarel Cohen 

Hasso Plattner Institute, University of Potsdam, Germany

Tobias Friedrich  

Hasso Plattner Institute, University of Potsdam, Germany

Martin Schirneck 

Hasso Plattner Institute, University of Potsdam, Germany

Abstract

We design f -edge fault-tolerant diameter oracles (f -FDO, or simply FDO if $f = 1$). For a given directed or undirected and possibly edge-weighted graph G with n vertices and m edges and a positive integer f , we preprocess the graph and construct a data structure that, when queried with a set F of edges, where $|F| \leq f$, returns the diameter of $G - F$. An f -FDO has stretch $\sigma \geq 1$ if the returned value \hat{D} satisfies $\text{diam}(G - F) \leq \hat{D} \leq \sigma \text{diam}(G - F)$.

For the case of a single edge failure ($f = 1$) in an unweighted directed graph, there exists an approximate FDO by Henzinger et al. [ITCS 2017] with stretch $(1 + \varepsilon)$, constant query time, space $O(m)$, and a combinatorial preprocessing time of $\tilde{O}(mn + n^{1.5} \sqrt{Dm/\varepsilon})$, where D is the diameter.

We present an FDO for directed graphs with the same stretch, query time, and space. It has a preprocessing time of $\tilde{O}(mn + n^2/\varepsilon)$, which is better for constant $\varepsilon > 0$. The preprocessing time nearly matches a conditional lower bound for combinatorial algorithms, also by Henzinger et al. With fast matrix multiplication, we achieve a preprocessing time of $\tilde{O}(n^{2.5794} + n^2/\varepsilon)$. We further prove an information-theoretic lower bound showing that any FDO with stretch better than $3/2$ requires $\Omega(m)$ bits of space. Thus, for constant $0 < \varepsilon < 3/2$, our combinatorial $(1 + \varepsilon)$ -approximate FDO is near-optimal in all parameters.

In the case of multiple edge failures ($f > 1$) in undirected graphs with non-negative edge weights, we give an f -FDO with stretch $(f + 2)$, query time $O(f^2 \log^2 n)$, $\tilde{O}(fn)$ space, and preprocessing time $\tilde{O}(fm)$. We complement this with a lower bound excluding any finite stretch in $o(fn)$ space.

Many real-world networks have polylogarithmic diameter. We show that for those graphs and up to $f = o(\log n / \log \log n)$ failures one can swap approximation for query time and space. We present an exact combinatorial f -FDO with preprocessing time $mn^{1+o(1)}$, query time $n^{o(1)}$, and space $n^{2+o(1)}$. When using fast matrix multiplication instead, the preprocessing time can be improved to $n^{\omega+o(1)}$, where $\omega < 2.373$ is the matrix multiplication exponent.

2012 ACM Subject Classification Theory of computation \rightarrow Shortest paths; Theory of computation \rightarrow Data structures design and analysis; Theory of computation \rightarrow Cell probe models and lower bounds; Theory of computation \rightarrow Pseudorandomness and derandomization

Keywords and phrases derandomization, diameter, distance sensitivity oracle, fault-tolerant data structure, space lower bound

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.18

Related Version *Full Version:* <https://arxiv.org/abs/2107.03485>

Funding *Davide Bilò:* This work was partially supported by the Research Grant FBS2016_BILO, funded by “Fondazione di Sardegna” in 2016.



© Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 18; pp. 18:1–18:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The diameter is one of the most fundamental graph parameters. It plays a particular significant role in the analysis of communication networks as the time to transmit a message to all nodes is strongly related with the diameter. Several lines of work have recently attacked the problem of computing the diameter in different settings. For example, Choudhary and Gold [16] constructed diameter spanners, which are subgraphs that approximately preserve the diameter of the original graph, Ancona et al. [6] developed algorithms for computing the diameter in dynamic scenarios and proved matching conditional lower bounds, and Bonnet [10] proved that, for any constant $\varepsilon > 0$, computing a $(7/4 - \varepsilon)$ -approximation of the diameter of a sparse graph n vertices and $m = n^{1+o(1)}$ edges requires $m^{4/3-o(1)}$ time, unless the Strong Exponential Time Hypothesis fails.

In this paper, we approach the diameter from the perspective of fault tolerance. A communication network may be subject to a small number of transient failures, and we want to quickly find out the new diameter without recomputing it from scratch. Therefore, we study the problem of constructing space-efficient data structures that can quickly report the diameter even if up to f edges fail in the graph. We refer to them as *f-edge fault-tolerant diameter oracles* (f -FDO, or simply FDO if $f = 1$). More precisely, given an undirected or directed and possibly edge-weighted graph G and a positive integer f , we want to construct an f -FDO that, when queried on a set F of up to f edges of G , returns a value \widehat{D} that is always at least as large the diameter of $G - F$, denoted by $\text{diam}(G - F)$. We say that an f -FDO has a *stretch* of $\sigma \geq 1$ (or that it is σ -approximate) if the value \widehat{D} returned by the oracle additionally satisfies $\text{diam}(G - F) \leq \widehat{D} \leq \sigma \text{diam}(G - F)$.

When designing f -FDOs one must find a good compromise between the following parameters: the stretch, the time needed to query the oracle, the size of the data structure, and the preprocessing time needed to build it. We focus particularly on space-optimal solutions, while keeping the query and preprocessing times low. For the case of a single edge failure in undirected edge-weighted graphs, there are two folklore FDOs known. One reports the exact diameter and has size $O(m)$, while the other takes $O(n)$ space, but guarantees only a stretch of 2. (more details are given in Section 2.1.) In a sense they mark the extreme points of a spectrum. It is natural to ask whether there are more trade-offs possible between the stretch and size of an FDO. More precisely, we pose the following question.

Question 1 – space vs. approximation trade-off. What is the minimum achievable size of an FDO for a given stretch σ ? To answer the question, we prove an information-theoretic lower bound. It shows that for undirected unweighted graphs and every (even non-constant) $1 \leq \sigma < 3/2$, every σ -approximate diameter oracle requires $\Omega(m)$ bits of space. The space lower bound also holds for the harder case of directed graphs. The size of the exact folklore FDO is thus optimal up to the size of a machine word. Moreover, we prove that the stretch 2 of the approximate FDO cannot be improved on weighted graphs while keeping $O(n)$ space.

► **Theorem 1.** *Any FDO with stretch $\sigma = \sigma(m) < 3/2$ must take $\Omega(m)$ bits of space on undirected graphs with m edges. The bound increases to $\sigma < 2$ if the graphs are edge-weighted.*

When we focus our attention on the preprocessing time, the exact FDO can be constructed in $\widetilde{O}(n^3)$ time¹ using the distance sensitivity oracle (DSO) of Bernstein and Karger [8]. Henzinger et al. [30] proved an essentially matching conditional lower bound for combinatorial²

¹ For a positive function $g(n, m, f)$, we use $\widetilde{O}(g)$ to denote $O(g \cdot \text{polylog}(n))$.

² The term “combinatorial algorithm” is not well-defined, and is often interpreted as not using any matrix multiplication. Arguably, combinatorial algorithms can be considered efficient in practice as the constants hidden in the matrix multiplication bounds are rather high.

algorithms. They assumed that any combinatorial algorithm requires $n^{3-o(1)}$ time to multiply two Boolean $n \times n$ matrices, known as the BMM conjecture. The restriction to combinatorial algorithms is crucial as the task is reducible to integer matrices and one can use fast matrix multiplication to solve it in $O(n^\omega)$ time, where $\omega < 2.37286$ is the matrix multiplication exponent [2]. Under the BMM conjecture, Henzinger et al. [30] showed that, for $0 < \varepsilon < 1/3$, any combinatorial preprocessing algorithm requires $n^{3-o(1)}$ time to build an FDO of stretch $(1 + \varepsilon)$, even if we allow $O(n^{2-\delta})$ query time for any constant $\delta > 0$.

They match this bound with an FDO with stretch $(1 + \varepsilon)$ and $O(1)$ query time that can be constructed in time $\tilde{O}(mn + n^{1.5} \sqrt{\text{diam}(G)} \cdot m/\varepsilon)$. Their oracle also reports the radius and vertex eccentricities in the presence of a single edge failure. Even on sparse graphs with $m = \tilde{O}(n)$ edges and constant diameter, the preprocessing time is $\tilde{O}(n^{2.5}/\sqrt{\varepsilon})$. For constant $\varepsilon > 0$, this is by a factor \sqrt{n} larger than the $\tilde{O}(mn)$ time needed to build the DSO of Bernstein and Karger [8]. It is interesting whether one can close the gap.

Question 2 – fast preprocessing time. Does there exist a combinatorial algorithm that constructs in $\tilde{O}(mn)$ time an FDO with stretch $(1 + \varepsilon)$ and constant query time? In addition, can one bypass the combinatorial lower bound by using fast matrix multiplication? We answer these questions affirmatively for the diameter case with the following theorem. The proof of the algebraic part uses the DSO presented very recently by Gu and Ren [28].

► **Theorem 2.** *For every unweighted directed graph and $\varepsilon > 0$, there exists a randomized combinatorial $(1 + \varepsilon)$ -approximate FDO that takes $O(m)$ space and has $\tilde{O}(mn + n^2/\varepsilon)$ preprocessing time and $O(1)$ query time. The returned values are correct w.h.p.³ Using fast matrix multiplication instead, one can construct the FDO in time $\tilde{O}(n^{2.5794} + n^2/\varepsilon)$.*

Note that, for any constant $0 < \varepsilon < 1/3$, our combinatorial $(1 + \varepsilon)$ -approximate combinatorial FDO from Theorem 2 is near-optimal with respect to all parameters. The $\Theta(m)$ space is near-optimal by Theorem 1, the query time is $\tilde{O}(1)$, and the $\tilde{O}(mn)$ preprocessing time comes within sub-polynomial factors of the conditional lower bound by Henzinger et al. [30]. Furthermore, when fast matrix multiplication is permitted, our algebraic preprocessing algorithm is even faster on dense graphs. However, our FDO is randomized.

Question 3 – derandomization. Can the construction of Theorem 2 be derandomized in the same asymptotic running time? We answer this question partially in that we derandomize the approximation part of our algorithm. When combined with the DSO of Bernstein and Karger [8] this gives a deterministic combinatorial FDO. For the derandomization, we adapt the framework of Alon, Chechik, and Cohen [3]. We identify a set of $O(n^{3/2})$ critical paths one needs to hit, and show how to compute them in $O(mn)$ time. It is then enough to let the folklore greedy algorithm compute a hitting set in $\tilde{O}(n^2)$ time.

It remains an open problem whether one can derandomize the algebraic approach, whose randomization stems solely from the DSO by Gu and Ren [28].

► **Theorem 3.** *For every unweighted directed graph and $\varepsilon > 0$, there exists a deterministic combinatorial $(1 + \varepsilon)$ -approximate FDO that takes $O(m)$ space and has $\tilde{O}(mn + n^2/\varepsilon)$ preprocessing time and $O(1)$ query time.*

³ An event occurs *with high probability* (w.h.p.) if it has probability at least $1 - n^{-c}$ for some $c > 0$.

Question 4 – space vs. approximation trade-off for multiple failures. Finally, we consider the case of multiple edge failures and examine similar questions. What is the minimum size for an exact, respectively, approximate, diameter oracle in the presence of up to f edge failures? We again prove an information-theoretic lower bound and show that for arbitrary finite stretch σ , any σ -approximation diameter oracle requires $\Omega(fn)$ bits of space, at least if the oracle can be queried also with sets F that contain non-edges.

► **Theorem 4.** *Suppose $f < n$. Any f -FDO with finite stretch that can be queried also for non-edges must take $\Omega(fn)$ bits of space on graphs with n vertices.*

We develop an efficient f -FDO whose space requirement almost matches the lower bound. Our result adapts and improves a construction by Bilò et al. [9]. Note that we use the \tilde{O} -notation to suppress polylogarithmic factors in n .

► **Theorem 5.** *For every undirected graph with non-negative edge weights, there exists a deterministic combinatorial $(f + 2)$ -approximate f -FDO that takes $\tilde{O}(fn)$ space and has $\tilde{O}(fm)$ preprocessing time and $\tilde{O}(f^2)$ query time.*

Real-world networks are often described as having a small diameter, dubbed as the “small world property” [35]. Many graph models used to analyze social and communication networks have provable polylogarithmic guarantees on the diameter, e.g. Chung-Lu graphs [17], hyperbolic random graphs [24], or the preferential attachment model [31]. We show that on graphs with low diameter one can swap approximation for query time even for multiple failures, while still retaining efficient preprocessing time and a low space requirement. To achieve this, we combine fault-tolerant trees that were introduced by Chechik et al. [13] with the random graphs of Weimann and Yuster [43].

► **Theorem 6.** *Let f be a positive integer and $\delta = \delta(n, m) > 0$ a real number. For every undirected unweighted graph with diameter at most $n^{\delta/f}/(f+1)$, there exists a randomized combinatorial f -FDO that takes $O(n^{2+\delta})$ space, has $O(2^f)$ query time, and with high probability $\tilde{O}(fmn^{1+\delta} + fn^{2+(2-1/f)\delta})$ preprocessing time. Using fast matrix multiplication instead, one can construct the FDO w.h.p. in time $\tilde{O}(fn^{\omega+\delta} + fn^{2+(2-1/f)\delta})$.*

If the diameter is in fact polylogarithmic and the number of failures is bounded by $f = o(\log n / \log \log n)$, we obtain the following corollary.

► **Corollary 7.** *Let $f = o(\log n / \log \log n)$. For every undirected graph with polylogarithmic diameter, there is an f -FDO that takes $n^{2+o(1)}$ space and has $n^{o(1)}$ query time. It can be preprocessed in time $mn^{1+o(1)}$, or algebraically in time $n^{\omega+o(1)}$. If f is constant, the preprocessing times are $\tilde{O}(mn)$, resp. $\tilde{O}(n^\omega)$, with $\tilde{O}(n^2)$ space and $\tilde{O}(1)$ query time.*

1.1 Related Work

We briefly review previous work on distance sensitivity oracles and diameter computation.

Distance sensitivity oracles. Distance oracles for all-pairs distances were introduced in a seminal paper by Thorup and Zwick [42]. Demetrescu et al. [19] extended the notion of distance oracles to the fault-tolerant setting in which either an edge or a vertex of a graph can fail (i.e., distance sensitivity oracles or DSOs). They showed that it is possible to preprocess a directed weighted graph in $\tilde{O}(mn^2)$ time to compute a data-structure of size $O(n^2 \log n)$ capable of answering distance queries in constant time. Bernstein and Karger [8] improved the preprocessing time to $\tilde{O}(mn)$ and Duan and Zhang [23] reduced the space to $O(n^2)$, which is asymptotically optimal.

Duan and Pettie [22] considered the more involved case of two failures and presented an oracle with $O(n^2 \log^3 n)$ size, $O(\log n)$ query time and polynomial construction time. Chechik et al. [13] presented a DSO of size $O(n^{2+o(1)})$ that supports up to $o(\log n / \log \log n)$ edge failures and guarantees a stretch of $(1 + \epsilon)$, for every constant $\epsilon > 0$. The approach has been recently extended to also handle vertex failures by Duan, Gu, and Ren [21].

The construction of DSOs have also been considered in the approximate regime [13]. Algebraic algorithms are known to improve the preprocessing times, if one is willing to employ fast matrix multiplication (for e.g., see [28, 12] and the references therein).

Diameter computation. The fastest known combinatorial algorithms (up to polylogarithmic factor) for both solving the all-pairs shortest paths (APSP) problem and the diameter problem, are the trivial ones with $\tilde{O}(mn)$ running time. There is extensive research on developing faster approximate APSP algorithms [7, 18, 33], as well as faster approximation algorithms for the diameter [14, 40]. For special classes of graphs, for example planar graphs, efficient exact algorithms for computing the diameter are known [25].

2 Preliminaries

We let $G = (V, E)$ denote the (possibly directed) base graph on n vertices and m edges. We tacitly assume that G is (strongly) connected, i.e., $m = \Omega(n)$. For a graph H , we denote by $V(H)$ the set of its vertices, and by $E(H)$ its edges. The (*closed*) *neighborhood* of a vertex $v \in V(H)$ is the set $N[v] = \{u \in V(H) \mid \{v, u\} \in E(H)\} \cup \{v\}$. Let P be a path in H , its *length* $|P|$ is the number of its edges. For any two vertices $x, y \in V(P)$, $P[x..y]$ is the subpath of P from x to y . For $s, t \in V(H)$, the *distance* $d_H(s, t)$ is the minimum length of an s - t -paths in H ; if s and t are disconnected, we set $d_H(s, t) = +\infty$. We drop the subscript when talking about the base graph G . The *eccentricity* of s is $\text{ecc}(s, H) = \max_{t \in V(H)} d_H(s, t)$ and the *diameter* is $\text{diam}(H) = \max_{s \in V(H)} \text{ecc}(s, H)$. Any graph distance can be stored in a single machine word on $O(\log n)$ bits. Unless explicitly stated otherwise, we measure the space complexity in the number of words. For a collection $F \subseteq \binom{V(H)}{2}$ of 2-sets of vertices (edges or non-edges), let $H - F$ be the graph obtained from H by removing all edges in F (graph H is not altered if $F \cap E(H) = \emptyset$). A *replacement path* $P_H(s, t, F)$ is a shortest path from s to t in $H - F$. Its length $d_H(s, t, F) = |P_H(s, t, F)|$ is the *replacement distance*. The *fault-tolerant diameter* of H with respect to F is the diameter of $H - F$.

For a positive integer f , an f -*fault-tolerant diameter oracle* (f -FDO) for the graph G is a data structure that reports, upon query F with $|F| \leq f$, the value $\text{diam}(G - F)$. For any $\sigma = \sigma(n, m, f) \geq 1$, such an oracle is σ -*approximate*, or has *stretch* σ , if it answers a query F with a value \hat{D} such that $\text{diam}(G - F) \leq \hat{D} \leq \sigma \cdot \text{diam}(G - F)$. In case of a single failure, we write FDO for 1-FDO and abbreviate $F = \{e\}$ to e . An f -*distance sensitivity oracle* (f -DSO) reports, upon query (s, t, F) with $|F| \leq f$, the replacement distance $d(s, t, F)$.

2.1 (Mostly) Known FDOs for Single Edge Failures

The first folklore FDO handles single edge failures in unweighted (directed or undirected) graphs. It has also been observed in [30]. The DSO of Bernstein and Karger [8] constructible in $\tilde{O}(mn)$ time and is able to report in constant time the exact distance of any pair of vertices in the presence of a single edge failure. With this one can construct the FDO by explicitly computing all the eccentricities $\text{ecc}(v, G - e)$, for every vertex v and every edge e of G , in $O(n^3)$ time. For a fixed vertex v , the m values $\text{ecc}(v, G - e)$ can be obtained in $O(n^2)$ time as follows. First compute a shortest paths tree T_v of G rooted at v . For each edge e that

is not in T_v , we have that $\text{ecc}(v, G - e) = \text{ecc}(v, G)$. For the tree-edges e in T_v , we use the DSO to compute $\text{ecc}(v, G - e)$ which is the maximum distance from v to any other vertex in $G - e$. Therefore, $\text{ecc}(v, G - e)$ can be computed by performing $n - 1$ queries, as there are $n - 1$ edges in T_v , we need $O(n^2)$ time. The fault-tolerant diameter $\text{diam}(G - e)$ is the maximum of the $\text{ecc}(v, G - e)$, it can be stored in $O(m)$ space with one entry for each edge e .

The second folklore FDO can only be used for undirected edge-weighted graphs. The FDO has stretch 2 and uses the fact that the diameter of the graph is intimately related to the eccentricity of *any* vertex. For an arbitrary v , we have that $\text{ecc}(v, G) \leq \text{diam}(G) \leq 2 \text{ecc}(v, G)$ as, by the triangle inequality, we can bound the distance between any two vertices $u, u' \in V$ by $d_G(u, u') \leq d_G(u, v) + d_G(v, u') \leq 2 \text{ecc}(v, G)$. The FDO again computes a shortest paths tree T rooted at a fixed source v and stores an array of length $n - 1$, corresponding to the edges of T . For every such edge e , one computes and stores $2 \text{ecc}(s, G - e)$. When queried with edge e , the FDO returns the stored value or, if e is not in the tree, the value $2 \text{ecc}(s, G)$. The size of this FDO is $O(n)$.

A maybe lesser-known way of building FDOs is via spanners. For any $\sigma > 0$, we say that a subgraph H of G is a *spanner* of stretch σ if, for every two vertices s, t of G , we have $d_H(s, t) \leq \sigma d_G(s, t)$. For every positive integer k , it is known how to construct a spanner H of G such that (a) H has a stretch of $2k - 1$ and (b) the size of H is $O(n^{1+1/k})$ [5]. Observe that for every edge $e = \{u, v\}$ that is in G but not in H , we have $d(u, v, e) \leq 2k - 1$. We now describe how spanners can be used to construct another easy oracle for undirected unweighted graphs whose stretch guarantee depends on both k and the inverse of $\text{diam}(G)$. This implies that the oracle already performs quite well for large-diameter graphs.

We construct such a spanner oracle with parameter k by first computing a spanner that satisfies (a) and (b). Then, we associate the value $\text{diam}(G - e)$ to each edge e in the spanner H and build a dictionary in which we store information about the edges of the spanner together with the corresponding associated values. Consider a query of edge e . If $e \in E(H)$ then we return the value associated with e ; otherwise, we return $\text{diam}(G) + 2(k - 1)$. The proof of the next lemma is deferred to the full version.

► **Lemma 8.** *For every positive integer k , the spanner oracle with parameter k has $O(n^{1+1/k})$ size, a constant query time, and a stretch of $1 + 2(k - 1)/\text{diam}(G)$.*

The result of Lemma 8 already implies the existence of sparse FDOs of $o(m)$ size and of stretch $\sigma < 3/2$ for sufficiently dense graphs with diameter strictly larger than 4. This does not contradict the lower bound of Theorem 1, but allows us to conclude that strong lower bounds on the size of FDOs for unweighted undirected graphs can only hold when the diameter of the input graph is bounded by a small constant.

3 Single Edge Failures

First, we treat single edge failures, $f = 1$. In this section, we assume the base graph G to be directed and present an $(1 + \varepsilon)$ -approximate fault-tolerant diameter oracle with space $O(m)$ and $O(1)$ query time. We give two variants, one is deterministic and combinatorial, the other randomized and algebraic. We then show that the space requirement is optimal up to the size of the machine word.

3.1 An $(1 + \varepsilon)$ -approximate FDO for Single Failures

We construct here the approximate FDO, thereby proving Theorem 2. Suppose we know for each $s, t \in V$ some shortest path $P(s, t)$ in G and additionally have access to a distance sensitivity oracle that, for any edge e , reports in constant time the replacement distances

$d(s, t, e)$ whenever needed. Clearly, $d(s, t, e)$ differs from the original graph distance only if e is on $P(s, t)$. To determine the diameters of *all* the graphs $G - e$, it is thus enough to query the DSO only for the edges on the shortest paths, which can be done in time $O(n^2 \cdot \text{diam}(G))$. We use approximation to avoid the cubic running time in case of a large diameter. For this, we randomly sample a small set B of so-called *pivots* and prove that it is enough to compute the replacement distances only between pairs from $B \times V$, instead of all pairs of vertices. Subsequently, we derandomize the pivot selection.

We fill in the details starting with the APSP computation in G and the preprocessing of the DSO. The combinatorial version uses a breath-first search from every vertex and the DSO of Bernstein and Karger [8], taking total time $\tilde{O}(mn)$. Alternatively, compute APSP algebraically and use the randomized DSO by Gu and Ren [28].⁴ APSP is computable in time $O(n^{2.575})$ on unweighted directed graphs with a variant of Zwick's algorithm [44, Corollary 4.5], this is in turn dominated by the $O(n^{2.5794})$ preprocessing time of the DSO [28]. After these computation, the distances $d(s, t)$, shortest paths $P(s, t)$ in G , and the replacement distances $d(s, t, e)$ are available to us (w.h.p., in the randomized case) with a constant query time per distance/path edge.

From here on out, the process for both variants is the same. Our fault-tolerant diameter oracle also allows non-edges to be queried, for which we return the original diameter $\text{diam}(G)$. To account for this, we store all edges in a static dictionary of size $O(m)$ that allows for worst-case constant look-up times after an $\tilde{O}(m)$ preprocessing [4, 29].⁵

Now fix a parameter $\varepsilon > 0$ for the approximation, possibly even depending on m, n . We initialize an array D indexed by the edges of G , all its cells hold the value $\text{diam}(G)$. Assume first that $\varepsilon \cdot \text{diam}(G) = O(\log n)$. For any two vertices s, t and edge e on the shortest path $P(s, t)$, we update $D[e]$ to the maximum of the previous value and $d(s, t, e)$. This takes $O(n^2 \text{diam}(G)) = \tilde{O}(n^2/\varepsilon)$ time. After all updates, the entry $D[e]$ stores the *exact* fault-tolerant diameter $\text{diam}(G - e)$ (possibly w.h.p.). For $\varepsilon \cdot \text{diam}(G) = \omega(\log n)$, we first give a randomized $(1+\varepsilon)$ -approximation and later derandomize it in Section 3.2. This yields the deterministic combinatorial algorithm of Theorem 3. The remaining use of randomness in the algebraic variant is due to the DSO by Gu and Ren [28].

To guard for the case that the failure of e disconnects the graph, we compute all *strong bridges* of G , that is, edges whose removal increases the number of strongly connected components, in time $O(m)$ with the algorithm by Italiano, Laura, and Santaroni [32]. For each strong bridge e , we set $D[e] = \infty$. To compute the other entries, we construct the set $B \subseteq V$ of pivots by randomly sampling every vertex independently with probability $C(\log n)/(\varepsilon \text{diam}(G))$ for a sufficiently large constant $C > 0$. A simple calculation using Chernoff bounds shows that $|B| = \tilde{O}(n/(\varepsilon \text{diam}(G)))$ w.h.p. Moreover, with high probability for all $s, t \in V$ and $e \in E$ such that $\varepsilon \text{diam}(G) < d(s, t, e) < \infty$, there exists a replacement path from s to t that avoids e and additionally contains a pivot from B . See [27, 41] for details. We update the entries of D in the same fashion as above, but now only use the (directed) distance $d(x, t, e)$ for all pivots $x \in B$ and vertices $t \in V$. In the end, we add $\varepsilon \text{diam}(G)$ to the value in $D[e]$. The array D is computable in time $O(m + n|B| \text{diam}(G)) = \tilde{O}(n^2/\varepsilon)$.

⁴ The DSO by Gu and Ren [28] is not path-reporting; if it were, we would not have to compute APSP. The fastest path-reporting algebraic DSO was given by Ren [38, 39] and can be constructed in time $O(n^{2.7233})$ on directed graphs, respectively in time $O(n^{2.6865})$ on undirected graphs.

⁵ The weak non-uniformity mentioned in [29], i.e., the need of compile-time constants depending on the word size, only holds if this size is $\omega(\log n)$, which is not the case for us.

We verify that $D[e]$ is an $(1+\varepsilon)$ -approximation of the fault-tolerant diameter $\text{diam}(G - e)$.

► **Lemma 9.** *We have $\text{diam}(G - e) \leq D[e] \leq (1+\varepsilon) \text{diam}(G - e)$ w.h.p.*

Proof. We can assume that $G - e$ is strongly connected as otherwise $D[e] = \infty = \text{diam}(G - e)$. The upper bound follows from $D[e] = \max_{x \in B, t \in V} d(x, t, e) + \varepsilon \text{diam}(G) \leq (1+\varepsilon) \text{diam}(G - e)$.

The main part consists of showing the lower bound $D[e] \geq \text{diam}(G - e)$. The idea is to prove the existence of a pivot $x \in B$ and vertex $t \in V$ whose replacement distance underestimates the fault-tolerant diameter by at most an additive term $\varepsilon \text{diam}(G)$, which we offset when computing $D[e]$. If $\text{diam}(G - e) \leq \varepsilon \text{diam}(G)$ (which can only happen for $\varepsilon \geq 1$), the lower bound holds vacuously as we have $D[e] \geq \varepsilon \text{diam}(G)$.

Let thus vertices $s, t \in V$ be such that $d(s, t, e) = \text{diam}(G - e) > \varepsilon \text{diam}(G)$. Since $G - e$ is strongly connected the diameter is finite and realized by some replacement path $P(s, t, e)$. In particular, we have $|P(s, t, e)| > \varepsilon \text{diam}(G)$. Let y be the unique vertex on $P(s, t, e)$ with $d(s, y, e) = \varepsilon \text{diam}(G)$. Recall that w.h.p. the set B hits *some* shortest path P' from s to y that avoids e . The path P' is not necessarily equal to the subpath $P(s, t, e)[s..y]$, but they have the same length $d(s, y, e)$. Substituting P' for $P(s, t, e)[s..y]$ therefore guarantees a replacement path from s to t that (w.h.p.) has a pivot $x \in B$ on its prefix of length $\varepsilon \text{diam}(G)$. For notational convenience, we use $P(s, t, e)$ to also denote this particular path.

The replacement distance from pivot x to target t satisfies $d(x, t, e) = |P(s, t, e)[x..t]| = |P(s, t, e)| - |P(s, t, e)[s..x]| \geq d(s, t, e) - \varepsilon \text{diam}(G)$. The entry $D[e]$ is also updated using the pivot x , whence $D[e] \geq d(x, t, e) + \varepsilon \text{diam}(G) \geq d(s, t, e) = \text{diam}(G - e)$. ◀

3.2 Derandomization

For the randomized combinatorial FDO, we had a preprocessing time of $\tilde{O}(mn + n^2/\varepsilon)$. The underlying APSP computation and the DSO are deterministic. We now derandomize the approximation part in the same asymptotic running time, proving Theorem 3. In Lemma 9, we used that the set B intersects at least one long replacement path from s to t exactly. We argue that it is in fact enough to hit the set of all vertices with distance at most $\varepsilon \text{diam}(G)$ from s in each strongly connected $G - e$. The pivot x does not need to be on any replacement path. The only assertion of Lemma 9 that is possibly in doubt is the lower bound $D[e] \geq \text{diam}(G - e)$. Let again s and t be such that $d(s, t, e) = \text{diam}(G - e)$ and let $x \in B$ be a pivot with $d(s, x, e) = d_{G - e}(s, x) \leq \varepsilon \text{diam}(G)$. Whenever $G - e$ is strongly connected, a replacement path $P(x, t, e)$ exists and, by the triangle inequality, we have $d(x, t, e) \geq d(s, t, e) - d(s, x, e) \geq d(s, t, e) - \varepsilon \text{diam}(G)$. The claim follows.

For the derandomization, we adopt the framework of Alon, Chechik and Cohen [3]. This involves efficiently finding a small set of critical paths such that hitting them ensures to hit each $(\varepsilon \text{diam}(G))$ -ball in the strongly connected $G - e$. If the critical paths are both short enough and few in numbers, it is then enough to compute the hitting set via the folklore greedy algorithm. In [3], it was sufficient to give a single set of critical paths. We generalize this to multiple sets, where the later-defined sets depend on the paths in the former.

Set $\ell = \min\{\varepsilon \text{diam}(G), \sqrt{n}\}$ and let r be an arbitrary vertex in G . We compute the in-tree $T_{\text{in}}(r)$, containing the shortest paths in G leading to r , with breath-first search. In the set \mathcal{P} , we collect, for each vertex s with $d(s, r) > \ell$, the path of $T_{\text{in}}(r)$ starting in s and having length ℓ . Let $P \in \mathcal{P}$ be a path with start vertex s and let $e \in E(P)$ be such that it is not a strong bridge. We compute the in-tree $T_{\text{in},e}(r)$ in $G - e$ rooted in r . Note that s has distance $d(s, r, e) \geq d(s, r) > \ell$ from the root in the tree. We add the corresponding path to the set \mathcal{P}_e . The original in-tree $T_{\text{in}}(r)$ contains only $n - 1$ edges, so all

trees can be computed in total time⁶ $O(mn)$. Moreover, there are at most $\ell + 1$ paths with starting vertex s . In total, we thus have $O(n\ell)$ paths each of length ℓ . A greedy algorithm computes a hitting set B for all paths in the \mathcal{P} and \mathcal{P}_e . It iteratively selects the vertex that is contained in the most yet unhit paths, it terminates in time $\tilde{O}(n\ell^2) = \tilde{O}(n^2)$ and produces a set of $|B| = \tilde{O}(n/\ell) = \tilde{O}(n/(\varepsilon \text{diam}(G)))$ pivots, see [3, 34]. We used the definition $\ell = \min\{\varepsilon \text{diam}(G), \sqrt{n}\}$ for both estimates. Finally, we add the root r to the set B to cover all paths in the trees that are shorter than ℓ .

► **Lemma 10.** *For each vertex $s \in V$ and edge e such that $G - e$ is strongly connected, there exists a pivot $x \in B$ with $d(s, x, e) \leq \varepsilon \text{diam}(G)$.*

Proof. If $d(s, r) \leq \varepsilon \text{diam}(G)$, we are done. Otherwise, let P be the prefix of length ℓ of the path from s to r in the tree $T_{\text{in}}(r)$, whence $P \in \mathcal{P}$. If P does not contain the edge e , it also exists in $G - e$ and the corresponding pivot $x \in B \cap V(P)$ satisfies $d(s, x, e) = d(s, x) \leq \ell \leq \varepsilon \text{diam}(G)$. If P contains e , then let instead $P' \in \mathcal{P}_e$ be the length- ℓ prefix of the path from s to r in $T_{\text{in},e}(r)$. Again, $x \in B \cap V(P')$ implies $d(s, x, e) \leq \varepsilon \text{diam}(G)$. ◀

3.3 Space Lower Bounds

Finally, we prove Theorem 1 thus showing that the space requirement of the FDOs in Theorems 2 and 3 is near-optimal provided that the stretch is $\sigma = \sigma(m, n) < 3/2$, that is, $\varepsilon < 1/2$. This even holds for the simpler task of computing the diameter in undirected graphs. For better exposition, we first show that any diameter oracle with such a stretch requires $\Omega(n^2)$ space on at least one n -vertex graph, which is, however, only tight for dense graphs. We then sparsify the construction to for an $\Omega(m)$ bound for graphs with m edges. Any σ -approximate FDO solves the promise problem of distinguishing, for each edge e , whether $G - e$ has diameter 2 or 3.

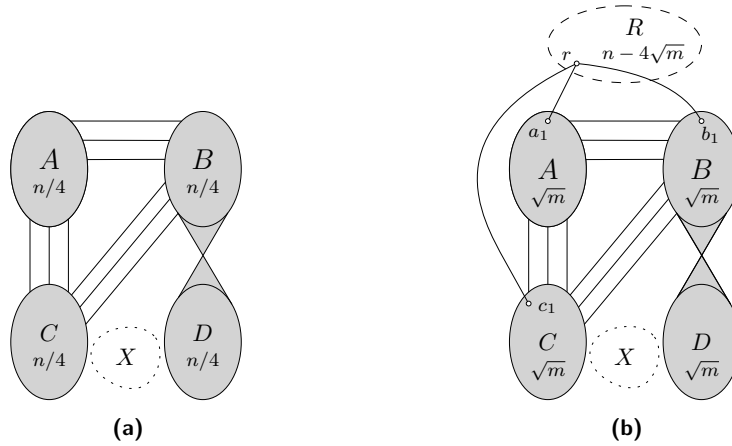
► **Lemma 11.** *There is a graph G on n vertices such that $G - e$ has diameter 2 or 3 for any $e \in E$. Any data structure that decides which one is the case must take $\Omega(n^2)$ bits of space.*

Proof. We give an incompressibility argument by encoding any binary $(n/4) \times (n/4)$ matrix X in the fault-tolerant diameters of G . No data structure can store this in $o(n^2)$ bits. The construction is illustrated in Figure 1a.

Without losing generality, n is divisible by 4, we can add up to three dummy vertices if needed. Split the vertex set equally into four groups A, B, C, D and let $a_1, \dots, a_{n/4}$ be an arbitrary numbering of the elements of A , same with the other groups. All groups are made into cliques and, for all $i \in [n/4]$, we make a_i, b_i , and c_i into a triangle. This results in matchings for the pairs (A, B) , (B, C) , and (A, C) , respectively. We further add edges so as to make (B, D) into a biclique. To encode the matrix X , we introduce the edge $\{c_i, d_j\}$ if and only if $X_{i,j} = 1$.

The graph G indeed has diameter 2 (even if X is the all-zeros matrix). Vertices a_i and b_j are joined by the path (a_i, a_j, b_j) —which by symmetry also holds for the other pairs of groups among A, B , or C —and the vertices a_i or c_i are connected to d_j via the paths (a_i, b_i, d_j) or (c_i, b_i, d_j) , respectively. Removing any edge increases the diameter by at most 1 since for any $e = \{u, v\}$ there exists a common neighbor in $w \in N[u] \cap N[v]$. This is clear inside the (bi-)cliques. For the matching edges, say $e = \{a_i, b_i\}$, we have $w = a_j, j \neq i$. Finally, for $e = \{c_i, d_j\}$ (if it exists), we have $w = b_i$.

⁶ For a single source, there are *randomized* algorithms known that compute the trees faster [11, 15, 26].



■ **Figure 1** Illustration of Lemma 11 **(a)** and of Lemma 12 **(b)**. The full ellipses A, B, C, D are cliques on the respective number of vertices, the dashed ellipse R is an independent set. The three parallel lines stand for matchings, the two crossed lines for a biclique. Edges encoding the binary matrix X run between C and D . Every vertex $r \in R$ is connected to $a_1 \in A$, $b_1 \in B$, and $c_1 \in C$.

We now prove that the graph $G - \{b_i, d_j\}$ has diameter 3 if and only if the edge $\{c_i, d_j\}$ is *not* present in G , that is, iff $X_{i,j} = 0$. When arguing the diameter above, edge $\{b_i, d_j\}$ was only needed for the paths (a_i, b_i, d_j) and (c_i, b_i, d_j) . Consider the neighborhoods of the three vertices in $G - \{b_i, d_j\}$, $N[a_i] = A \cup \{c_i, d_i\}$, $N[c_i] = C \cup \{a_i, b_i\} \cup \{d_k \mid X_{i,k} = 1\}$, and $N[d_j] = D \cup (B \setminus \{b_i\}) \cup \{c_k \mid X_{k,j} = 1\}$. If $X_{i,j} = 1$, then the neighborhoods intersect, namely in c_i , keeping the diameter at 2. If, however, $X_{i,j} = 0$, then $N[a_i] \cap N[d_j] = \emptyset$ and the diameter increases to 3. ◀

We now refine the result to give a better bound for sparse graphs. Note that a logarithmic gap remains between Lemma 12 and Theorem 2 since we lower bound the space at $\Omega(m)$ bits while the FDO takes this many words.

► **Lemma 12.** *There is a graph G with m edges such that $G - e$ has diameter 2 or 3 for any edge $e \in E$. A data structure that decides which one is the case must take $\Omega(m)$ bits of space.*

Proof. The main weakness of the construction in Lemma 11 is that it requires $\Omega(n^2)$ edges inside the cliques. As it turns out, this is not necessary and we can sparsify the graph G as long as we keep its diameter at 2. Figure 1b shows the idea of the sparsification.

Let m' be a parameter to be fixed later. We now store a binary $\sqrt{m'} \times \sqrt{m'}$ matrix X . Split the vertices into five groups, where A, B, C, D each contain $\sqrt{m'}$ vertices and R the remaining $n - 4\sqrt{m'}$. The edges among vertices in A through D are the same as in Lemma 11. Each vertex in R has degree 3 and is connected to a_1, b_1 , and c_1 . The graph G has $4\binom{\sqrt{m'}}{2} + 3\sqrt{m'} + m' + |\{(i, j) \mid X_{i,j} = 1\}| + 3(n - 4\sqrt{m'}) = O(m')$ edges. We fix the parameter m' such that the total number of edges is m . If needed, we introduce additional edges among vertices in R without affecting the result.

Note that the eccentricity of any vertex in $r \in R$ is 2 (even if R is not an independent set). Vertex a_i is reached via the path (r, a_1, a_i) , similar for the vertices in B and C , the ones in D are reached via b_1 . Moreover, for any edge involving r , say $\{r, a_1\}$, we have $b_1 \in N[r] \cap N[a_1]$. Therefore, the proof that G has diameter 2, $G - e$ has diameter 2 or 3, and $G - \{b_i, d_j\}$ has diameter 3 iff $X_{i,j} = 0$ is almost exactly as in Lemma 11. The sole difference is the case in which the edge $\{b_1, d_j\}$ fails since this may also increase the eccentricity of r . This is settled by observing that the neighborhood $N[r] = \{r, a_1, b_1, c_1\}$ in $G - \{b_1, d_j\}$ intersects $N[d_j]$ iff $X_{1,j} = 1$. To accommodate all possible matrices X , we require $\Omega(m') = \Omega(m)$ bits. ◀

The same construction shows that for edge-weighted graphs there is no $(2-\varepsilon)$ -approximate FDO, for any $\varepsilon = \varepsilon(m) > 0$, with space $o(m)$. In more detail, we choose an $\varepsilon' > 0$ small enough so that $\varepsilon' < 2\varepsilon/(1-\varepsilon)$ and give weight ε' to all matching edges as well as the edges incident to vertices in R , all other edges are weighted 2. One can verify that $\text{diam}(G) = 2 + \varepsilon'$ and the fault-tolerant diameter $\text{diam}(G - \{b_i, d_j\})$ remains at that value iff $\{c_i, d_j\}$ is present, it raises to $4 + \varepsilon'$ otherwise. The bound on the stretch cannot be improved as shown by the trivial FDO discussed in the introduction, which gives a 2-approximation in $O(n)$ space.

4 Multiple Edge Failures

We now turn to multiple edge failures. Recall that in the fault-tolerant setting the maximum number f of failures is known in advance, and stretch, space, preprocessing, and query time usually depend on f . In this section, we first prove the following lemma. Let $\alpha = \alpha(m, n)$ denote the inverse Ackermann function.

► **Lemma 13** (Theorem 5 with explicit logarithmic factors). *For every undirected graph with non-negative edge weights, there exists a deterministic combinatorial $(f+2)$ -approximate f -FDO that takes $O(fn \log^2 n)$ space and has $O(fm\alpha + fn \log^3 n)$ preprocessing time and $O(f^2 \log^2 n)$ query time. For $f = 1$, the size of the oracle is $O(n)$, the preprocessing time $O(m\alpha + n \log n)$, and the query time is constant.*

Bilò et al. [9] designed an $(2f+1)$ -approximate single-source f -DSO. That means, the oracle processes an undirected graph G with non-negative edge weights and a distinguished source s , and, upon query (t, F) with $|F| \leq f$, it returns $d(s, t, F)$. The oracle can be built in $O(fm\alpha + fn \log^3 n)$ time, has size $O(fn \log^2 n)$, and answers queries in $O(f^2 \log^2 n)$ time. In principle we can modify the oracle so as, when queried with the set F , it returns twice the eccentricity of s in the graph $G - F$. This would clearly allow us to construct an f -FDO of stretch $2 \cdot (2f+1)$. We show that the same oracle construction, but with a better query algorithm, allows us to develop an f -FDO of stretch $f+2$.

We let $w(e)$ denote the weight of the edge $e \in E$. The length of a path is now defined as the sum of its edge weights; the definitions of distance and diameter are adjusted accordingly. The oracle in [9] first computes a shortest path tree T of G rooted at the source s and uses it to re-weight all the edges of G . The new weight function w' assigns weight of 0 to each edge of T and weight $w'(e) = d(s, x) + w(e) + d(y, s)$ to any other edge $e = \{x, y\}$. When queried with (t, F) , the oracle computes a spanning forest T_F of $G - F$ w.r.t. the new weight function w' in $O(f^2 \log^2 n)$ time. Let $k = |F \cap E(T)|$. The oracle replaces the k failing edges in $F \cap E(T)$ with a minimum-weight set of edges in $G - F$ w.r.t. to w' , say E_F , whose addition to $T - F$ forms a spanning forest of $G - F$.⁷ The obtained forest T_F is then used to estimate the distance from s to t in $G - F$. We reuse a nice property proven in [9].

► **Lemma 14** (Bilò et al. [9]). *T_F is a minimum spanning forest of $G - F$ w.r.t. w' .*

Our query algorithm works as follows. Let tree T be rooted at s and $F \cap E(T) = \{f_1, \dots, f_k\}$ with $k \leq f$ the edges in T that are also in F . Let T_0, \dots, T_k denote the $k+1$ subtrees of $T - F$, and r_i the root of the subtree T_i . W.l.o.g., we assume $r_0 = s$. We use

⁷ This is done by computing, for each unordered pair $\phi = (T', T'')$ of connected components of $T - F$, the minimum-weight edge w.r.t. w' , say e_ϕ , that has one endpoint in T' and the other endpoint in T'' . Then, the set E_F is computed in $O(f^2)$ time using any time-efficient algorithm for computing a minimum spanning tree of an auxiliary graph in which each of the connected components of $T - F$ is modelled by a vertex and the edge between the unordered pair $\phi = (T', T'')$ of $T - F$ has a weight equal to $w'(e_\phi)$. The authors of [9] design a data structure that is able to retrieve, for each pair $\phi = (T', T'')$ of connected components of $T - F$, the edge e_ϕ in $O(\log^2 n)$ time.

f_1, \dots, f_k to compute the roots r_1, \dots, r_k in $O(k)$ time. We then build a forest T' on $k+1$ new vertices v_0, \dots, v_k , where v_i represents T_i . The forest T' contains an edge $\{v_i, v_j\}$ iff E_F contains an edge e with one end point in $V(T_i)$ and the other in $V(T_j)$. Obviously, if T' is not connected, then we can simply certify that $\text{diam}(G-F) = \infty$. So, we assume that T' is a tree. We root T' at v_0 and denote by e_i the edge that joins v_i with its parent $p(v_i)$. We compute the value $\Delta = \max_{1 \leq i \leq k} w'(e_i) - d(s, r_i)$ and output $\widehat{D} = f\Delta + 2 \cdot \max_{t \in V} d(s, t)$. The time needed for the query algorithm is dominated by the computation of T in time $O(f^2 \log^2 n)$ as all the new operations can be performed in $O(f^2)$ time. Observe that $\max_{t \in V} d(s, t)$ is independent of F and can be precomputed in time $O(n)$.

For a single failure, $f = 1$, the query time can be reduced to $O(1)$. In fact, for each edge e of T , it is enough to precompute the minimum weight edge of $E(G) \setminus E(T)$, w.r.t. weight function w' , that crosses the cut induced by $T - e$. This, a.k.a. the sensitivity analysis problem of a minimum spanning tree, can be solved in $O(m \log \alpha)$ time on a graph with m edges [37]. We show in the remainder that \widehat{D} is an $(f+2)$ -approximation of $\text{diam}(G-F)$. The proof of the following lemma can be found in the full version.

► **Lemma 15.** *We have that $\text{diam}(G-F) \geq \Delta$.*

We now prove the approximation with the help of Lemma 15.

► **Lemma 16.** *The value \widehat{D} satisfies $\text{diam}(G-F) \leq \widehat{D} \leq (f+2) \text{diam}(G-F)$.*

Proof. Again, we only need to prove anything if T' is connected, which implies that T_F is connected. By Lemma 15, we have that $\text{diam}(G-F) \geq \Delta$. Moreover, $\text{diam}(G-F) \geq \text{diam}(G) \geq \max_{t \in V} d(s, t)$. The value \widehat{D} returned by the query algorithm satisfies $\widehat{D} \leq f\Delta + 2 \max_{t \in V} d(s, t) \leq (f+2) \text{diam}(G-F)$. It remains to show that $\widehat{D} \geq \text{diam}(G-F)$. We prove the latter by verifying that, for any two vertices x and y , $\widehat{D} \geq d(x, y, F)$ holds.

Let r_x and r_y be the roots of the subtrees of $T-F$ that contain x and y , respectively. It is possible that $r_x = r_y$. Let $r_{p(i)}$ denote the root of the tree of $T-F$ that corresponds to the parent vertex $p(v_i)$ in T' . Consider the subgraph of T_F consisting of the edges of the paths in T_F between the following pairs of vertices: (a) $r_{p(i)}$ and r_i for every i , (b) x and r_x , (c) y and r_y . The subgraph contains a path from x to y since T_F is connected. Therefore, the replacement distance $d(x, y, F)$ is upper bounded by the total weight of the subgraph. The path in T_F between r_x and x has length at most $\max_{t \in V} d(s, t)$ as r_x is an ancestor of x in the shortest path tree T rooted at s ; same for r_y and y . Finally, for any $i > 0$, let $e_i = \{x_i, y_i\}$ be the edge in E_F that caused the addition of the edge $(v_i, p(v_i))$ in T' . W.l.o.g., we assume that x_i (resp., y_i) is a vertex of the tree of $T-F$ represented by v_i (resp., $p(v_i)$) in T' . The path from r_i to $r_{p(i)}$ in $G-F$ has length at most $d(r_i, x_i) + w(e_i) + d(y_i, r_{p(i)}) \leq d(r_i, x_i) + w(e_i) + d(y_i, s) + d(s, r_i) - d(s, r_i) = w'(e_i) - d(s, r_i) \leq \Delta$. Therefore, $d(x, y, F) \leq k\Delta + 2 \max_{t \in V} d(s, t) \leq f\Delta + 2 \max_{t \in V} d(s, t) = \widehat{D}$. ◀

4.1 Exact f -FDO for Low Diameter

We show that one can swap approximation for query time in low-diameter graphs, namely, with diameter at most $n^{\delta/f}/(f+1)$ for arbitrary $\delta = \delta(m, n) > 0$. This is summarized in Theorem 6. The case $f=1$ is solved like in Section 3.1 only that there is no need for approximation here as the diameter is small enough to process all pairs of vertices in time $O(n^{2+\delta})$. We thus assume $f \geq 2$. We adapt a space-saving technique introduced by Chechik et al. [13]. In a bird's-eye view, we construct a recursion tree $T(s, t)$ of size $O(n^\delta)$ for each pair of vertices s and t . It contains all *relevant* replacement distances $d(s, t, F)$ for sets F with up to f failures. We then show how we can simulate the search for $\text{diam}(G-F)$ in the $O(n^2)$ trees in total time $O(2^f)$.

Afek et al. [1, Theorem 1] showed that if G is undirected, then any shortest path in $G - F$, with $|F|$, is a concatenation of at most $|F| + 1$ shortest paths in G . The condition on the diameter and $|F| \leq f$ ensure that every path below has length at most $(f+1) \cdot \text{diam}(G) \leq n^{\delta/f}$.

Assume we have access to a path-reporting f -DSO. That means, upon query (s, t, F) , the oracle either certifies that $d(s, t, F) = \infty$, i.e., s and t are disconnected in $G - F$, or reports the replacement distance and a shortest s - t -path in $G - F$. The preprocessing time of the combinatorial version is assumed to be $\tilde{O}(f m n^{1+\delta})$ with a $\tilde{O}(f n^{(1-1/f)\delta} + |P|) = \tilde{O}(f n^{(1-1/f)\delta})$ query time w.h.p. reporting path P . Here, we used the assumption $f \geq 2$, whence $|P| \leq n^{\delta/f} = \tilde{O}(f n^{(1-1/f)\delta})$. Alternatively, we have algebraic preprocessing in time $\tilde{O}(f n^{\omega+\delta})$. We show how to obtain the oracle in the full version, using an idea of Weimann and Yuster [43] with a more refined analysis of the query time.

Fix two vertices s and t . We construct the tree $T(s, t)$ recursively. Each node in the tree is associated with a set $F' \subseteq \binom{V}{2}$ containing $f' = |F'| \leq f$ possible failures. We have $F' = \emptyset$ in the root. Upon creation, the node queries the assumed oracle with (s, t, F') and holds the returned path $P(s, t, F')$, if any. If $f' = f$ or s and t are disconnected in $G - F'$, the node is a leaf. Otherwise, it has $d(s, t, F')$ many children, one for each edge of $e \in E(P(s, t, F'))$ of the path. The respective child is associated with the set $F' \cup \{e\}$.

The tree indeed has at least one node for every distinct replacement distance $d(s, t, F)$ with $|F| \leq f$. To see this, let F', F be two sets with $F' \subseteq F \subseteq \binom{V}{2}$. Clearly, we have $d(s, t, F') \leq d(s, t, F)$, but $d(s, t, F') < d(s, t, F)$ can only hold if $F \setminus F'$ contains an edge of the path $P(s, t, F')$ in the node associated with F' . The fan-out of each node is at most $n^{\delta/f}$, the height of the tree is f . For all $s, t \in V$, the trees thus have $O(n^{2+\delta})$ nodes in total and can be constructed with that many queries to the f -DSO in time $\tilde{O}(f n^{2+(2-1/f)\delta})$.

Consider the following naive algorithm to handle a query to the f -FDO for the fault-tolerant diameter $\text{diam}(G - F)$. Each tree $T(s, t)$ is searched individually starting in the root. The processing of a node depends on the associated set F' . If it is a leaf or the set $F \setminus F'$ is disjoint from the replacement path $P(s, t, F')$, then we return the length $d(s, t, F')$ of the path; otherwise, we recurse on all children associated with $F' \cup \{e\}$ for all edges $e \in (F \setminus F') \cap E(P(s, t, F'))$. By the argument as above, the maximum over all reported distances is indeed $\max_{s, t \in V; F' \subseteq F} d(s, t, F') = \text{diam}(G - F)$. This approach can be improved significantly by aggregating the values $\{d(s, t, F')\}_{s, t \in V}$ already at construction.

Observe that we never query the underlying f -DSO with a set F' that contains non-edges. We prepare a hash table H whose entries are indexed by subsets of E of size at most f . For every query (s, t, F') we compare the returned replacement distance with the value $H[F']$. If no such entry exists, we initialize it with $d(s, t, F')$; else, we update it to $\max\{H[F'], d(s, t, F')\}$. The final table has size $O(n^{2+\delta})$ and we discard the trees. The table H is constructible w.h.p. in time $O(n^{2+\delta})$, guaranteeing constant query time [20, 36]. However, to simulate the naive algorithm for the query F to the f -FDO, we have to check $H[F']$ for all $O(2^f)$ subsets $F' \subseteq F$ as we do not know which ones were used during construction.

4.2 Space Lower Bound

We conclude with the space lower bound of Theorem 4. It rules out any finite stretch in $o(fn)$ space for an arbitrary number f of failures. We use the fact that an f -FDO with finite stretch is able to decide whether the edges in F are a cut-set of the graph.

Assume for now that f is even. Let k be the largest integer such that $fk + 1 \leq n$. We construct a graph G as follows. It has vertices c, v_1, \dots, v_{fk} as well as $n - fk - 1$ auxiliary vertices. Define $E_i = \{\{v_i, v_j\} \mid 1 \leq |i - j| \leq f/2\}$. The edge set of G is $\bigcup_{i=1}^{fk} E_i$ together with all possible edges $\{c, u\}$, including to the auxiliaries. In other words, G consists of a star

centered at c with $n - 1$ leaves, and leaves v_i, v_j are joined by an edge iff their indices have difference at most $f/2$. Let set \mathcal{G} contain all spanning subgraphs of G that retain at least all star edges incident to c . Since $|E_i| = f$, there are $|\mathcal{G}| = 2^{(f-1)f/2} = 2^{\Omega(fn)}$ such subgraphs.

Let H be any subgraph in \mathcal{G} . For $i \neq j$ with $|i - j| \leq f/2$, define the set $F_{i,j} = (E_i \setminus \{v_i, v_j\}) \cup \{c, v_i\}$. Note that $F_{i,j}$ may contain non-edges. We have $|F_{i,j}| = f$ and evidently $\{v_i, v_j\}$ is present in H iff $H - F_{i,j}$ is connected. Any two f -FDOs for graphs in \mathcal{G} thus differ in at least one bit. For odd values $f \geq 3$, we emulate this using $f - 1$ failures.

For the remaining case $f = 1$, we use a different construction. W.l.o.g., n is even, connecting a single excess vertex to some other vertex in the graph is immaterial. The graph G contains two parallel paths P_1 and P_2 , each on $n/2$ vertices, respectively numbered from 1 to $n/2$. The graph also contains a matching M in which the i -th vertex of P_1 is matched with the i -th vertex of P_2 . Let \mathcal{G} be the set of all spanning subgraphs that have at least all the edges of P_1 and M . We have $|\mathcal{G}| = 2^{(n/2)-1} = 2^{\Omega(n)}$. Let $H \in \mathcal{G}$ and define e_i , with $i < n/2$, be the edge of P_1 between the i -th and $(i+1)$ -th vertices. The corresponding edge of P_2 is present in H if and only if $H - e_i$ is connected.

References

- 1 Yehuda Afek, Anat Bremler-Barr, Haim Kaplan, Edith Cohen, and Michael Merritt. Restoration by Path Concatenation: Fast Recovery of MPLS Paths. *Distributed Computing*, 15:273–283, 2002. doi:10.1007/s00446-002-0080-6.
- 2 Josh Alman and Virginia Vassilevska Williams. A Refined Laser Method and Faster Matrix Multiplication. In *Proceedings of the 32nd Symposium on Discrete Algorithms (SODA)*, pages 522–539, 2021. doi:10.1137/1.9781611976465.32.
- 3 Noga Alon, Shiri Chechik, and Sarel Cohen. Deterministic Combinatorial Replacement Paths and Distance Sensitivity Oracles. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 12:1–12:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.12.
- 4 Noga Alon and Moni Naor. Derandomization, Witnesses for Boolean Matrix Multiplication and Construction of Perfect Hash Functions. *Algorithmica*, 16:434–449, 1996. doi:10.1007/BF01940874.
- 5 Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On Sparse Spanners of Weighted Graphs. *Discrete and Computational Geometry*, 9:81–100, 1993. doi:10.1007/BF02189308.
- 6 Bertie Ancona, Monika Henzinger, Liam Roditty, Virginia Vassilevska Williams, and Nicole Wein. Algorithms and Hardness for Diameter in Dynamic Graphs. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 13:1–13:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.13.
- 7 Surender Baswana and Telikepalli Kavitha. Faster Algorithms for All-pairs Approximate Shortest Paths in Undirected Graphs. *SIAM Journal on Computing*, 39:2865–2896, 2010. doi:10.1137/080737174.
- 8 Aaron Bernstein and David R. Karger. A Nearly Optimal Oracle for Avoiding Failed Vertices and Edges. In *Proceedings of the 41st Symposium on Theory of Computing (STOC)*, pages 101–110, 2009. doi:10.1145/1536414.1536431.
- 9 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Multiple-Edge-Fault-Tolerant Approximate Shortest-Path Trees. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 18:1–18:14, 2016. doi:10.4230/LIPIcs.STACS.2016.18.
- 10 Édouard Bonnet. 4 vs 7 Sparse Undirected Unweighted Diameter is SETH-hard at Time $n^{4/3}$. In *Proceedings of 48th International Colloquium on Automata, Languages, and Programming, (ICALP)*, 2021. To appear.

- 11 Shiri Chechik and Sarel Cohen. Near Optimal Algorithms for the Single Source Replacement Paths Problem. In *Proceedings of the 30th Symposium on Discrete Algorithms (SODA)*, pages 2090–2109, 2019. doi:10.1137/1.9781611975482.126.
- 12 Shiri Chechik and Sarel Cohen. Distance Sensitivity Oracles with Subcubic Preprocessing Time and Fast Query Time. In *Proceedings of the 52nd Symposium on Theory of Computing (STOC)*, pages 1375–1388, 2020. doi:10.1145/3357713.3384253.
- 13 Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. $(1 + \epsilon)$ -Approximate f -Sensitive Distance Oracles. In *Proceedings of the 28th Symposium on Discrete Algorithms (SODA)*, pages 1479–1496, 2017. doi:10.1137/1.9781611974782.96.
- 14 Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert E. Tarjan, and Virginia Vassilevska Williams. Better Approximation Algorithms for the Graph Diameter. In *Proceedings of the 25th Symposium on Discrete Algorithms (SODA)*, pages 1041–1052, 2014. doi:10.1137/1.9781611973402.78.
- 15 Shiri Chechik and Ofer Magen. Near Optimal Algorithm for the Directed Single Source Replacement Paths Problem. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 81:1–81:17, 2020. doi:10.4230/LIPIcs.ICALP.2020.81.
- 16 Keerti Choudhary and Omer Gold. Extremal Distances in Directed Graphs: Tight Spanners and Near-Optimal Approximation Algorithms. In *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*, pages 495–514, 2020. doi:10.1137/1.9781611975994.30.
- 17 Fan Chung and Linyuan Lu. The Average Distances in Random Graphs with Given Expected Degrees. *Proceedings of the National Academy of Sciences*, 99:15879–15882, 2002. doi:10.1073/pnas.252631999.
- 18 Edith Cohen and Uri Zwick. All-Pairs Small-Stretch Paths. *Journal of Algorithms*, 38:335–353, 2001. doi:10.1006/jagm.2000.1117.
- 19 Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for Distances Avoiding a Failed Node or Link. *SIAM Journal on Computing*, 37:1299–1318, 2008. doi:10.1137/S0097539705429847.
- 20 Martin Dietzfelbinger, Anna R. Karlin, Kurt Mehlhorn, Friedhelm Meyer auf der Heide, Hans Rohnert, and Robert E. Tarjan. Dynamic Perfect Hashing: Upper and Lower Bounds. *SIAM Journal on Computing*, 23:738–761, 1994. doi:10.1137/S0097539791194094.
- 21 Ran Duan, Yong Gu, and Hanlin Ren. Approximate Distance Oracles Subject to Multiple Vertex Failures. In *Proceedings of the 32nd Symposium on Discrete Algorithms (SODA)*, pages 2497–2516, 2021. doi:10.1137/1.9781611976465.148.
- 22 Ran Duan and Seth Pettie. Dual-Failure Distance and Connectivity Oracles. In *Proceedings of the 20th Symposium on Discrete Algorithms (SODA)*, pages 506–515, 2009. URL: <https://dl.acm.org/citation.cfm?id=1496770.1496826>.
- 23 Ran Duan and Tianyi Zhang. Improved Distance Sensitivity Oracles via Tree Partitioning. In *Proceedings of the 15th Algorithms and Data Structures Symposium (WADS)*, pages 349–360, 2017. doi:10.1007/978-3-319-62127-2_30.
- 24 Tobias Friedrich and Anton Krohmer. On the Diameter of Hyperbolic Random Graphs. *SIAM Journal on Discrete Mathematics*, 32:1314–1334, 2018.
- 25 Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi Diagrams on Planar Graphs, and Computing the Diameter in Deterministic $\tilde{O}(n^{5/3})$ Time. In *Proceedings of the 29th Symposium on Discrete Algorithms (SODA)*, pages 495–514, 2018. doi:10.1137/1.9781611975031.33.
- 26 Fabrizio Grandoni and Virginia Vassilevska Williams. Improved Distance Sensitivity Oracles via Fast Single-Source Replacement Paths. In *Proceedings of the 53rd Symposium on Foundations of Computer Science (FOCS)*, pages 748–757, 2012. doi:10.1109/FOCS.2012.17.
- 27 Fabrizio Grandoni and Virginia Vassilevska Williams. Faster Replacement Paths and Distance Sensitivity Oracles. *ACM Transaction on Algorithms*, 16:15:1–15:25, 2020. doi:10.1145/3365835.

- 28 Yong Gu and Hanlin Ren. Constructing a Distance Sensitivity Oracle in $O(n^{2.5794}M)$ Time. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2021. To appear.
- 29 Torben Hagerup, Peter Bro Miltersen, and Rasmus Pagh. Deterministic Dictionaries. *Journal of Algorithms*, 41:69–85, 2001. doi:10.1006/jagm.2001.1171.
- 30 Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional Hardness for Sensitivity Problems. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 26:1–26:31, 2017. doi:10.4230/LIPIcs.ITCS.2017.26.
- 31 Remco van der Hofstad. *Random Graphs and Complex Networks*, volume 1 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, UK, 2016. doi:10.1017/9781316779422.
- 32 Giuseppe F. Italiano, Luigi Laura, and Federico Santaroni. Finding Strong Bridges and Strong Articulation Points in Linear Time. *Theoretical Computer Science*, 447:74–84, 2012. doi:10.1016/j.tcs.2011.11.011.
- 33 Telikepalli Kavitha. Faster Algorithms for All-Pairs Small Stretch Distances in Weighted Graphs. *Algorithmica*, 63:224–245, 2012. doi:10.1007/s00453-011-9529-y.
- 34 Valerie King. Fully Dynamic Algorithms for Maintaining All-Pairs Shortest Paths and Transitive Closure in Digraphs. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*, pages 81–91, 1999. doi:10.1109/SFFCS.1999.814580.
- 35 Jon M. Kleinberg. Navigation in a Small World. *Nature*, 406:845–845, 2000. doi:10.1038/35022643.
- 36 Rasmus Pagh and Flemming Friche Rodler. Cuckoo Hashing. *Journal of Algorithms*, 51:122–144, 2004. doi:10.1016/j.jalgor.2003.12.002.
- 37 Seth Pettie. Sensitivity Analysis of Minimum Spanning Trees in Sub-Inverse-Ackermann Time. *Journal of Graph Algorithms and Applications*, 19:375–391, 2015. doi:10.7155/jgaa.00365.
- 38 Hanlin Ren. Improved Distance Sensitivity Oracles with Subcubic Preprocessing Time. In *Proceedings of the 28th European Symposium on Algorithms (ESA)*, pages 79:1–79:13, 2020. doi:10.4230/LIPIcs.ESA.2020.79.
- 39 Hanlin Ren. Improved Distance Sensitivity Oracles with Subcubic Preprocessing Time. *CoRR*, abs/2007.11495, 2020. ArXiv preprint. Full version of [38]. arXiv:2007.11495.
- 40 Liam Roditty. Approximating the Diameter. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 116–117. Springer, New York City, NY, USA, 2016. doi:10.1007/978-1-4939-2864-4_566.
- 41 Liam Roditty and Uri Zwick. Replacement Paths and k Simple Shortest Paths in Unweighted Directed Graphs. *ACM Transaction on Algorithms*, 8:33:1–33:11, 2012. doi:10.1145/2344422.2344423.
- 42 Mikkel Thorup and Uri Zwick. Approximate Distance Oracles. In *Proceedings on 33rd Symposium on Theory of Computing (STOC)*, pages 183–192, 2001. doi:10.1145/380752.380798.
- 43 Oren Weimann and Raphael Yuster. Replacement Paths and Distance Sensitivity Oracles via Fast Matrix Multiplication. *ACM Transactions on Algorithms*, 9:14:1–14:13, 2013. doi:10.1145/2438645.2438646.
- 44 Uri Zwick. All Pairs Shortest Paths Using Bridging Sets and Rectangular Matrix Multiplication. *Journal of the ACM*, 49:289–317, 2002. doi:10.1145/567112.567114.

ω -Forest Algebras and Temporal Logics

Achim Blumensath 

Masaryk University, Brno, Czech Republic

Jakub Lédl 

Masaryk University, Brno, Czech Republic

Abstract

We use the algebraic framework for languages of infinite trees introduced in [4] to derive effective characterisations of various temporal logics, in particular the logic EF (a fragment of CTL) and its counting variant cEF.

2012 ACM Subject Classification Theory of computation \rightarrow Logic

Keywords and phrases forest algebras, temporal logics, bisimulation

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.19

Funding *Achim Blumensath*: Work supported by the Czech Science Foundation, grant No. GA17-01035S.

Jakub Lédl: Work supported by the Czech Science Foundation, grant No. GA17-01035S.

1 Introduction

Among the many different approaches to language theory, the algebraic one seems to be particularly convenient when studying questions of expressive power. While algebraic language theories for word languages (both finite and infinite) were already fully developed a long time ago, the corresponding picture for languages of trees, in particular infinite ones, is much less complete. Seminal results contributing to such an algebraic framework for languages of infinite trees were provided by the group of Bojańczyk [7, 8] with one article considering languages of *regular* trees only, and one considering languages of *thin* trees. The first complete framework that could deal with arbitrary infinite trees was provided in [2, 3]. Unfortunately, it turned out to be too complicated and technical for applications. Recently, two new general frameworks have been introduced [1, 4] which seem to be more satisfactory: one is based on the notion of a *branch-continuous tree algebra*, while the other uses *regular tree algebras*. The first one seems to be more satisfactory from a theoretical point of view, while the second one is more useful for applications, in particular for characterisation results.

In this article we concentrate on the approach based on regular tree algebras from [4] which seems to be emerging as the standard. The goal is to apply the framework to a few test cases and to see how well it performs for its intended purpose. While the definition of a regular tree algebra (given in Section 2 below) is a bit naïve and seems circular at first sight, it turns out that it is sufficient to guarantee the properties we need for applications: one can show that (i) the class of regular tree algebras forms a pseudo-variety and that (ii) every regular tree language has a syntactic algebra, which is in fact a regular tree algebra. By general category-theoretic results, such as those from [6] or [5], this implies that there exists a Reiterman type theorem for such algebras, i.e., the existence of equational characterisations for sub-pseudo-varieties. This is precisely what is needed for a characterisation theorem.

Unfortunately progress on an algebraic theory of infinite trees has been rather slow since matters have turned out to be significantly more complicated than the case of words or finite trees. Hence, every step of progress is very welcome. For instance, the recent paper [11] characterises the languages of infinite trees that are recognised by algebras of bounded growth. The applications we are looking at in the present paper concern certain temporal logics,



© Achim Blumensath and Jakub Lédl;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 19; pp. 19:1–19:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in particular, the logic EF and its counting variant cEF, and we aim to derive decidable algebraic characterisations for them using our algebraic framework. Note that Bojańczyk and Idziaszek have already provided a decidable characterisation for EF in [7], but their result is only partially algebraic. They prove that a regular language is definable in EF if, and only if, the language is bisimulation-invariant and its syntactic algebra satisfies a certain equation, but they were not able to provide an algebraic characterisation of bisimulation invariance. Due to our more general algebraic framework we are able to fill this gap below.

We start in the next section with a short overview of the algebraic framework from [4]. We have to slightly modify this material since it was originally formulated in the setting of ranked trees while, when looking at temporal logics, it is more natural to consider unranked trees and forests. The remainder of the article contains our various characterisation results. In Section 3 we derive an algebraic characterisation of bisimulation-invariance, the result missing in [7]. Then, in Section 4, we turn to our main result and present characterisations for the logic cEF and some of its fragments, including a new and complete characterisation of the logic EF.

2 Forest algebras

The main topic of this article are languages of (possibly infinite) forests and the logics defining them. Before introducing the algebras we will use to recognise such languages, let us start by fixing some notation and conventions. Although our main interest is in unranked forests, we will use a more general version that combines the ranked and the unranked cases. As we will see below (cf. Theorem 3.1), the ability to use ranks will increase the expressive power of equations for our algebras considerably. Thus, we will work with *ranked sets*, i.e., sets where every element a is assigned an *arity* $\text{ar}(a)$. Formally, we consider such sets as families $A = (A_m)_{m < \omega}$, where A_m is the set of all elements of A of arity m . Functions between ranked sets then take the form $f = (f_m)_{m < \omega}$ with $f_m : A_m \rightarrow B_m$.

We will consider (unranked, finitely branching, possibly infinite) forests where each vertex is labelled by an element of a given ranked set A and each edge is labelled by a natural number with the restriction that, if a vertex is labelled by an element of arity m , the numbers labelling the outgoing edges must be less than m . If an edge $u \rightarrow v$ is labelled by the number k , we will call v a *k-successor* of u . Note that a vertex may have several k -successors, or none at all. We assume that the roots of a forest are ordered from left to right, as are all the k -successors of a given vertex v , while we impose no ordering between a k -successor and an l -successor, for $k \neq l$. We write $\mathbb{F}_0 A$ for the set of all such A -labelled forests. (We shall explain the index 0 further below.) We write $\text{dom}(s)$ for the set of vertices of a forest $s \in \mathbb{F}_0 A$, and we will usually identify s with the function $s : \text{dom}(s) \rightarrow A$ that maps vertices to their labels. We denote the empty forest by 0 and the disjoint union of two forests s and t by $s + t$ (where the roots of t are added after those of s). We will frequently use term notation to denote forests such as

$$a(b + c, 0, b) + b,$$

which denotes a forest with two components: the first one consisting of a root labelled by an element a of arity 3 which has two 0-successors labelled b and c , no 1-successor, and one 2-successor; the second component consists of a singleton with label b .

We use the symbol \preceq for the forest ordering where the roots are the minimal elements and the leaves the maximal ones. For a forest s , we denote by $s|_v$ the subtree of s attached to the vertex v . The *successor forest* of v in s is the forest obtained from $s|_v$ by removing the root v .

For a natural number n , set $[n] := \{0, \dots, n-1\}$. An *alphabet* is a finite (unranked) set Σ of symbols. If we use an alphabet in a situation such as $\mathbb{F}_0\Sigma$ where a ranked set is expected, we will consider each symbol in Σ as having arity 1. Thus, for us a *forest language over an alphabet* Σ will be a set $L \subseteq \mathbb{F}_0\Sigma$ consisting of the usual unranked forests. (The power to have elements of various arities is useful when writing down algebraic equations, but it is rather unnatural when considering languages defined by temporal logics.) We denote by Σ^* the set of all finite words over Σ , by Σ^ω the set of infinite words, and $\Sigma^\infty := \Sigma^* \cup \Sigma^\omega$. A *family of (word, forest, ...) languages* is a function \mathcal{K} mapping each alphabet Σ to a class $\mathcal{K}[\Sigma]$ of (word, forest, ...) languages over Σ .

Our algebraic framework to study forest languages is built on the notion of an Eilenberg–Moore algebra for a monad. To keep category-theoretical prerequisites at a minimum we will give an elementary, self-contained definition. The basic idea is that, in the same way we can view the product of a semigroup as an operation turning a sequence of semigroup elements into a single element, we view the product of a forest algebra as an operation turning a given forest that is labelled with elements of the algebra into a single element. The material in this section is taken from [4] with minor adaptations to accommodate the fact that we are dealing with unranked forests instead of ranked trees. Proofs can also be found in [5], although in a much more general setting. We start by defining which forests we allow in this process.

► **Definition 2.1.**

- (a) We denote by \mathbb{F} the functor mapping a ranked set A to the ranked set $\mathbb{F}A = (\mathbb{F}_m A)_m$ where $\mathbb{F}_m A$ consists of all $(A \cup \{x_0, \dots, x_{m-1}\})$ -labelled forests such that
- the new labels x_0, \dots, x_{m-1} have arity 0,
 - each label x_i appears at least once, but only finitely many times, and
 - no root is labelled by an x_i .
- (b) The *singleton function* $\text{sing} : A \rightarrow \mathbb{F}A$ maps a label a of arity m to the forest $a(x_0, \dots, x_{m-1})$.
- (c) The *flattening function* $\text{flat} : \mathbb{F}\mathbb{F}A \rightarrow \mathbb{F}A$ takes a forest $s \in \mathbb{F}\mathbb{F}A$ and maps it to the forest $\text{flat}(s)$ obtained by assembling all forests $s(v)$, for $v \in \text{dom}(s)$, into a single large forest. This is done as follows. For every vertex of $s(v)$ that is labelled by a variable x_k , we take the disjoint union of all forests labelling the k -successors of v and substitute them for x_k . This is done simultaneously for all $v \in \text{dom}(s)$ and all variables in $s(v)$ (see Figure 1 for an example.) ┘

Now we can define a forest algebra to be a set A equipped with a product $\mathbb{F}A \rightarrow A$.

► **Definition 2.2.**

- (a) An ω -forest algebra $\mathfrak{A} = \langle A, \pi \rangle$ consists of a ranked set A and a function $\pi : \mathbb{F}A \rightarrow A$ satisfying the following two axioms:

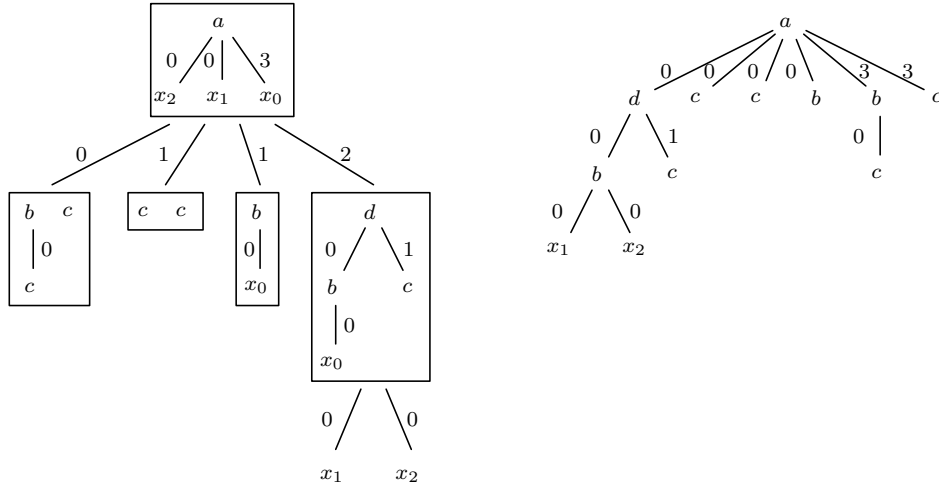
$$\text{the associative law } \pi \circ \mathbb{F}\pi = \pi \circ \text{flat} \quad \text{and} \quad \text{the unit law } \pi \circ \text{sing} = \text{id}.$$

We will denote forest algebras by fraktur letters \mathfrak{A} and their universes by the corresponding roman letter A . We will usually use the letter π for the product, even if several algebras are involved.

- (b) A *morphism* of ω -forest algebras is a function $\varphi : \mathfrak{A} \rightarrow \mathfrak{B}$ that commutes with the products in the sense that $\pi \circ \mathbb{F}\varphi = \varphi \circ \pi$. ┘

► **Remark.**

- (a) In the following we will simplify terminology by dropping the ω and simply speaking of *forest algebras*. But note that, strictly speaking, this name belongs to the kind of algebras introduced by Bojańczyk and Walukiewicz in [10].



■ **Figure 1** The flattening operation.

- (b) One can show that the functor \mathbb{F} together with the two natural transformations flat and sing forms what is called a *monad* in category theory. In this terminology, we can define forest algebras as *Eilenberg-Moore algebras* for this monad.
- (c) Note that a forest algebra $\mathfrak{A} = \langle A, \pi \rangle$ contains a monoid $\langle A_0, +, 0 \rangle$ (called the *horizontal monoid*) and an ω -semigroup $\langle A_1, A_0, \cdot \rangle$ (the *vertical ω -semigroup*), whose operations are derived from the product π . For instance, the vertical product $a \cdot b$, for $a, b \in A_1$, is formed as the product $\pi(s)$, where s consists of a root labelled a , an internal vertex labelled b , and a leaf labelled by the variable x_0 .
- (d) The reason why we do not allow forests where some root is labelled by a variable x_k is that an infinite product of such forests is not always defined. For instance, multiplying an infinite sequence of forests of the form $x_0 + a$ would create a forest with infinitely many components, which is not allowed.

Sets of the form $\mathbb{F}A$ can be equipped with a canonical forest algebra structure by using the flattening operation $\text{flat} : \mathbb{F}\mathbb{F}A \rightarrow \mathbb{F}A$ for the product. By general category-theoretical considerations it follows that algebras of this form are exactly the *free* forest algebras (generated by A). In this article we consider *forest languages* over an alphabet Σ as subsets $L \subseteq \mathbb{F}_0\Sigma$. Such a language is *recognised* by a morphism $\eta : \mathbb{F}\Sigma \rightarrow \mathfrak{A}$ of forest algebras if $L = \eta^{-1}[P]$ for some $P \subseteq A_0$.

Example. Let $\Sigma := \{a, b\}$. We can recognise the language $L \subseteq \mathbb{F}_0\Sigma$ of all forests s containing at least one occurrence of the letter a as follows. Let \mathfrak{A} be the algebra consisting of two elements 0_m and 1_m , for each arity m , where the product π maps a forest $s \in \mathbb{F}_m A$ to 1_n if at least one vertex is labelled by 1_n , for some n . Otherwise, s is mapped to 0_m . Then $L = \varphi^{-1}(1_0)$ where the morphism $\varphi : \mathbb{F}_0\Sigma \rightarrow \mathfrak{A}$ is defined by $\varphi(a) := 1_1$ and $\varphi(b) := 0_1$. (As $\mathbb{F}\Sigma$ is freely generated by the set $\{a, b\}$, this determines φ for all inputs.)

In analogy to the situation with word languages we would like to have a theorem stating that a forest language is regular if, and only if, it is recognised by a morphism into some finite forest algebra. But this statement is wrong for two reasons. The first one is that every forest algebra with at least one element of positive arity has elements of every arity and, thus, is infinite. (For instance, given $a \in A_1$, we obtain an element $a(x_0 + \dots + x_{n-1}) \in A_n$ of arity n). To fix this, we have to replace the property of being finite by that of having only finitely many elements of each arity. We call such algebras *finitary*.

But even if we modify the statement in this way it still fails since one can find finitary forest algebras recognising non-regular languages. (An example for tree languages is given by Bojańczyk and Klin in [9].) Therefore we have to restrict our class of algebras. A simple way to do so is given by the class of (locally) regular algebras introduced in [4] where all of the following results are taken from (again in the case of trees instead of forests).

► **Definition 2.3.** Let \mathfrak{A} be a forest algebra.

- (a) A subset $C \subseteq A$ is *regularly embedded* if, for every element $a \in A$, the preimage $\pi^{-1}(a) \cap \mathbb{F}C$ is a regular (i.e., automaton recognisable) language over C .
- (b) \mathfrak{A} is *locally regular* if every finite subset is regularly embedded.
- (c) \mathfrak{A} is *regular* if it is finitary, finitely generated, and locally regular. ┘

The definition of a regular forest algebra is not very enlightening. We refer the interested reader to [4] for a purely algebraic (but much more complicated) characterisation.

► **Theorem 2.4.** Let $L \subseteq \mathbb{F}_0\Sigma$ be a forest language. The following statements are equivalent.

- (1) L is regular (i.e., automaton recognisable).
- (2) L is recognised by a morphism into a locally regular forest algebra.
- (3) L is recognised by a morphism into a regular forest algebra.

(The reason why we introduce two classes is that locally regular algebras enjoy better closure properties, while the regular ones are more natural as recognisers of languages.) One can show (see [4]) that the (locally) regular algebras form a pseudo-variety in the sense that locally regular algebras are closed under quotients, subalgebras, finite products, and directed colimits, while regular algebras are closed under quotients, finitely generated subalgebras, finitely generated subalgebras of finite products, and so-called “rank-limits”. More important for our current purposes is the existence of syntactic algebras and the fact that these are always regular.

► **Definition 2.5.** Let $L \subseteq \mathbb{F}\Sigma$ be a forest language.

- (a) The *syntactic congruence* of L is the relation

$$s \sim_L t \quad \text{:iff} \quad p[s] \in L \Leftrightarrow p[t] \in L, \quad \text{for every context } p,$$

where a *context* is a $(\Sigma \cup \{\square\})$ -labelled forest (where \square is a new symbol of the same arity as s and t) and $p[s]$ is the forest obtained from p by replacing each vertex labelled by \square by the forest s .

- (b) The *syntactic algebra* of L is the quotient $\mathfrak{S}(L) := \mathbb{F}\Sigma / \sim_L$. ┘

► **Theorem 2.6.** The syntactic algebra $\mathfrak{S}(L)$ of a regular forest language L exists, it is regular, and it is the smallest forest algebra recognising L . Furthermore, $\mathfrak{S}(L)$ can be computed given an automaton for L .

Regarding the last statement of this theorem, we should explain what we mean by computing a forest algebra. Since forest algebras have infinitely many elements, we cannot simply compute the full multiplication table. Instead, we say that a regular forest algebra \mathfrak{A} is computable if, given a number $n < \omega$, we can compute a list $\langle \mathcal{A}_a \rangle_{a \in A_n}$ of automata such that \mathcal{A}_a recognises the set $\pi^{-1}(a) \cap \mathbb{F}C$, for some fixed set C of generators.

3 Bisimulation

To illustrate the use of syntactic algebras let us start with a simple warm-up exercise: we derive an algebraic characterisation of bisimulation invariance. This example also explains why algebras with elements of higher arities are needed (this is the reason Bojańczyk and Idziaszek [7], whose framework supported only arity 1, had to leave such a characterisation as an open problem).

Recall that a *bisimulation* between two forests s and t is a binary relation $Z \subseteq \text{dom}(s) \times \text{dom}(t)$ such that $\langle u, v \rangle \in Z$ implies that

- $s(u) = t(v)$ and,
- for every k -successor u' of u , there is some k -successor v' of v with $\langle u', v' \rangle \in Z$ and vice versa.

Two trees are *bisimilar* if there exists a bisimulation between them that relates their roots. More generally, two forests are bisimilar if every component of one is bisimilar to some component of the other. A language L of forests is *bisimulation-invariant* if $s \in L$ implies $t \in L$, for every forest t bisimilar to s .

► **Theorem 3.1.** *A forest language $L \subseteq \mathbb{F}_0\Sigma$ is bisimulation-invariant if, and only if, the syntactic algebra $\mathfrak{S}(L)$ satisfies the following equations:*

$$\begin{aligned} c + c &= c, & a(x_0 + x_0) &= a(x_0), \\ c + d &= d + c, & a(x_0 + x_1 + x_2 + x_3) &= a(x_0 + x_2 + x_1 + x_3), \end{aligned}$$

for all $a \in S_1(L)$ and $c, d \in S_0(L)$.

Proof. Let $\eta : \mathbb{F}\Sigma \rightarrow \mathfrak{S}(L)$ be the syntactic morphism mapping a forest to its \sim_L -class.

(\Rightarrow) Given elements $c, d \in S_0(L)$, we fix forests $s \in \eta^{-1}(c)$ and $t \in \eta^{-1}(d)$. If L is bisimulation-invariant, we have

$$p[s] \in L \quad \text{iff} \quad p[s + s] \in L \quad \text{and} \quad p[s + t] \in L \quad \text{iff} \quad p[t + s] \in L,$$

for every context p . Consequently, $s \sim_L s + s$ and $s + t \sim_L t + s$, which implies that $c = c + c$ and $c + d = d + c$.

The remaining two equations are proved similarly. Fix $a \in S_1(L)$ and $s \in \eta^{-1}(a)$. Setting $s' := s(x_0 + x_0)$, bisimulation-invariance of L implies that

$$p[s] \in L \quad \text{iff} \quad p[s'] \in L, \quad \text{for every context } p.$$

Consequently $s \sim_L s'$ and $a(x_0) = \eta(s) = \eta(s') = a(x_0 + x_0)$.

Similarly, for $t := s(x_0 + x_1 + x_2 + x_3)$ and $t' := s(x_0 + x_2 + x_1 + x_3)$, we have

$$p[t] \in L \quad \text{iff} \quad p[t'] \in L, \quad \text{for every context } p.$$

Hence, $t \sim_L t'$ and $a(x_0 + x_1 + x_2 + x_3) = a(x_0 + x_2 + x_1 + x_3)$.

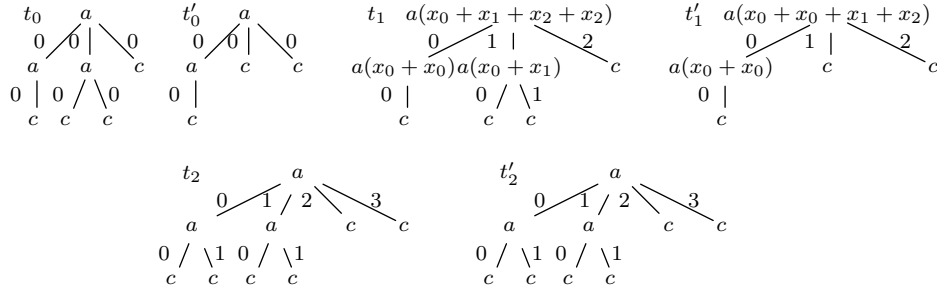
(\Leftarrow) Suppose that $\mathfrak{S}(L)$ satisfies the four equations above and let s and s' be bisimilar forests. We claim that $\eta(s) = \eta(s')$, which implies that $s \in L \Leftrightarrow s' \in L$.

Fix a bisimulation relation $Z \subseteq \text{dom}(s) \times \text{dom}(s')$. W.l.o.g. we may assume that Z only relates vertices on the same level of the respective forests and that it only relates vertices whose predecessors are also related. (If not, we can always remove the pairs not satisfying this condition without destroying the fact that Z is a bisimulation.) Let \approx be the equivalence relation on $\text{dom}(s) \cup \text{dom}(s')$ generated by Z .

We will transform the forests s and s' in several steps while preserving their value under η until both forests are equal. (Note that each of these steps necessarily modifies the given forest at every vertex.) An example of this process can be found in Figure 2. The first step consists in translating the problem into the algebra $\mathfrak{S}(L)$. We define two new forests $t_0, t'_0 \in \mathbb{F}_0S(L)$ with the same domains as, respectively, s and s' and the following labelling. If $v \in \text{dom}(s)$ has the 0-successors u_0, \dots, u_{n-1} , we set

$$t_0(v) := \eta(s(v))(x_0 + \dots + x_{n-1})$$

and we make u_i an i -successor of v in t_0 . We obtain t'_0 from s' in the same way. By associativity it follows that $\pi(t_0) = \eta(s)$ and $\pi(t'_0) = \eta(s')$.



■ **Figure 2** Transforming bisimilar forests.

Next we make the shapes of the forests t_0 and t'_0 the same. Let t_1 and t'_1 be the forests with the same domains as t_0 and t'_0 and the following labelling. For every vertex v of t_0 with successors u_0, \dots, u_{n-1} and labelling

$$t_0(v) = a(x_0 + \dots + x_{n-1}),$$

we set

$$t_1(v) := a(x_0 + \dots + x_0 + \dots + x_{n-1} + \dots + x_{n-1}),$$

where each variable x_i is repeated m_i times and the numbers m_i are determined as follows. Let M be some number such that, for every $i < n$, no vertex $v' \approx v$ has at more than M successors u' with $u' \approx u_i$. (Note that there are only finitely many such vertices.) We choose the constants m_i such that

$$\sum_{k \in U_i} m_k = M, \quad \text{where } U_i := \{k < n \mid u_k \approx u_i\}.$$

We obtain the forest t'_1 in the same way from t'_0 . By the top right equation in the statement of the theorem, the value of the product is not affected by this modification. Hence, $\pi(t_1) = \pi(t_0)$ and $\pi(t'_1) = \pi(t'_0)$.

Finally, let t_2 and t'_2 be the unravelling of, respectively, t_1 and t'_1 , i.e., the forest where for every vertex v with successors u_0, \dots, u_{n-1} and label

$$t_1(v) = a(x_0 + \dots + x_0 + \dots + x_{n-1} + \dots + x_{n-1}),$$

we set

$$t_2(v) := a(x_0 + \dots + x_k + \dots + x_l + \dots + x_m)$$

(where we number the variables from left-to-right, e.g., $a(x_0 + x_0 + x_1 + x_2 + x_2)$ becomes $a(x_0 + x_1 + x_2 + x_3 + x_4)$), and we duplicate each attached subforest a corresponding number of times such that the value of the product does not change. We do the same for t'_2 .

We have arrived at a situation where, for each component r of the forests t_2 , there is some component r' of t'_2 that differs only in the ordering of successors, but not in their number. Consequently, there exists a bijection $\sigma : \text{dom}(t) \rightarrow \text{dom}(r')$ such that, for a vertex v of r with successors u_0, \dots, u_{n-1} ,

$$r'(v) = r(v)(x_{\sigma_v(0)} + \dots + x_{\sigma_v(n-1)}),$$

where the function $\sigma_v : [n] \rightarrow [n]$ is chosen such that $\sigma(u_i)$ is the $\sigma_v(i)$ -successor of $\sigma(v)$.

Let \hat{r} be the tree obtained from r as follows. For a vertex v with successors u_0, \dots, u_{n-1} and labelling

$$r(v) = a(x_0 + \dots + x_{n-1}),$$

we set

$$\hat{r}(v) := a(x_{\sigma_v(0)} + \dots + x_{\sigma_v(n-1)}),$$

and we reorder the attached subtrees accordingly. By associativity and the bottom right equation, this does not change the value of the product. It follows that $\hat{r} = r'$. Consequently, $\pi(r) = \pi(r')$.

We have shown that, for every component of t_0 there is some component of t'_0 with the same product. Therefore, we can write

$$\pi(t_0) = a_0 + \dots + a_{m-1} \quad \text{and} \quad \pi(t'_0) = b_0 + \dots + b_{n-1}$$

where the sets $\{a_0, \dots, a_{m-1}\}$ and $\{b_0, \dots, b_{n-1}\}$ coincide. Using the equations $c + c = c$ and $c + d = d + c$ we can therefore transform $\pi(t_0)$ into $\pi(t'_0)$. Consequently,

$$\eta(s) = \pi(t_0) = \pi(t'_0) = \eta(s').$$

As η recognises L it follows that $s \in L \Leftrightarrow s' \in L$, as desired. \blacktriangleleft

Note that we immediately obtain a decision procedure for bisimulation-invariance from this theorem, since we can compute the syntactic algebra and check whether it satisfies the given set of equations.

► **Corollary 3.2.** *It is decidable whether a given regular language L is bisimulation-invariant.*

4 The Logic cEF

Let us now proceed to the main result of this article: a characterisation of the temporal logic cEF. For simplicity, the following definition of its semantics only considers forests instead of arbitrary transition systems.

► **Definition 4.1.**

(a) *Counting* EF, cEF for short, has two kinds of formulae: *tree formulae* and *forest formulae*, which are inductively defined as follows.

- Every forest formula is a finite boolean combination of formulae of the form $E_k \varphi$ where k is a positive integer and φ a tree formula.
- Every tree formula is a finite boolean combination of (i) forest formulae and (ii) formulae of the form P_a , for $a \in \Sigma$.

To define the semantics we introduce a satisfaction relation \models_f for forest formulae and one \models_t for tree formulae. In both cases boolean combinations are defined in the usual way. For a tree t , we define

$$\begin{aligned} t \models_t P_a & \quad \text{: iff} \quad \text{the root of } t \text{ has label } a, \\ t \models_t \varphi & \quad \text{: iff} \quad t' \models_f \varphi, \quad \text{for a forest formula } \varphi, \text{ where } t' \text{ denotes the successor} \\ & \quad \quad \quad \text{forest of the root of } t. \end{aligned}$$

For a forest s , we define

$$\begin{aligned} s \models_f E_k \varphi & \quad \text{: iff} \quad \text{there exist at least } k \text{ vertices } v, \text{ distinct from the roots, such that} \\ & \quad \quad \quad s|_v \models \varphi. \end{aligned}$$

- (b) For $k, m < \omega$, we denote by cEF_k the fragment of cEF that uses only operators E_l where $l \leq k$, and cEF_k^m is the fragment of cEF_k where the nesting depth of the operators E_l is restricted to m . For $k = 1$, we set $\text{EF} := \text{cEF}_1$ and $\text{EF}^m := \text{cEF}_1^m$. \square

The following is our main theorem. Before giving the statement a few technical remarks are in order. In the equations below we make use of the ω -power a^ω of an element $a \in A_1$ (which is the infinite vertical product $aaa\dots$), and the *idempotent power* a^π (which is the defined as $a^\pi = a^n$ for the minimal number n with $a^n a^n = a^n$). For the horizontal semigroup we use multiplicative notation instead: $n \times a$ for $a + \dots + a$ and $\pi \times a$ for $n \times a$ with n as above.

When writing an ω -power of an element of arity greater than one, we need to specify with respect to which variable we take the power. We use the notation a^{ω_i} to indicate that the variable x_i should be used. Note that, when using several ω -powers like in $(a(x_0, (b(x_0, x_1))^{\omega_1}))^{\omega_0}$, the intermediate term after resolving the inner power can be a forest with infinitely many occurrences of the variable x_0 . But after resolving the outer ω -power, we obtain a forest without variables, i.e., a proper element of $\mathbb{F}_0 A$. Consequently, the equations below are all well-defined. Finally, to keep notation light we will frequently write x instead of x_0 , if this is the only variable present.

► **Theorem 4.2.** *A forest language $L \subseteq \mathbb{F}_0 \Sigma$ is definable in the logic cEF_k if, and only if, the syntactic algebra $\mathfrak{S}(L)$ satisfies the following equations:*

$$\begin{array}{ll}
c + d = d + c & (a(x) + b(x))^\omega = (ab(x))^\omega \\
(ab)^\pi = b(ab)^\pi & (a(x) + c)^\omega = (a(x + c))^\omega \\
a^\omega + a^\omega = a^\omega & (a(x + c + c))^\omega = (a(x + c))^\omega \\
(abb')^\omega = (ab'b)^\omega & [a(b(x_0, x_1))^{\omega_1}]^{\omega_0} = [ab(x_0, x_0)]^{\omega_0} \\
(aab)^\omega = (ab)^\omega & [a(x + bc + c)]^\omega = [a(x + bc)]^\omega \\
\\
a_n(c, \dots, c) + (k - n) \times c = a_n(c, \dots, c) + (k - n + 1) \times c, \\
[a(x + (a(k \times x))^\pi(c))]^\omega = k \times (a(k \times x))^\pi(c)
\end{array}$$

for all $a, b, b' \in S_1(L)$, $c, d \in S_0(L)$, $a_n \in S_n(L)$, and $n \leq k$.

No attempt was made to simplify the above axioms. While having a simpler description would of course be nice, the importance of this result lies in the facts that (i) an equational axiomatisation exists; that (ii) the equations can be checked algorithmically; and (iii) that our framework was sufficient to derive them.

We defer the proof to the appendix. Let us here concentrate on some of the consequences instead.

► **Corollary 4.3.** *For fixed k , it is decidable whether a given regular language L is cEF_k -definable.*

For the logic cEF , where the value of k is not bounded, a similar result can now be derived as a simple corollary. The basic argument is contained in the following lemma.

► **Lemma 4.4.** *Given a forest algebra \mathfrak{A} that is generated by $A_0 \cup A_1$, we can compute a number K such that, if \mathfrak{A} satisfies the equations of Theorem 4.2 for some value of k , it satisfies them for $k = K$.*

Proof. Set $K := m_0^{2m_1} + m_0$ where $m_0 := |A_0|$ and $m_1 := |A_1|$. By assumption there is some number k for which \mathfrak{A} satisfies the equations of Theorem 4.2. W.l.o.g. we may assume that $k \geq K$. The only two equations depending on k are

19:10 ω -Forest Algebras and Temporal Logics

$$(1)_k \quad a_n(c, \dots, c) + (k - n) \times c = a_n(c, \dots, c) + (k - n + 1) \times c$$

$$(2)_k \quad [a(x + (a(k \times x))^\pi(c))]^\omega = k \times (a(k \times x))^\pi(c)$$

We have to show that \mathfrak{A} also satisfies $(1)_K$ and $(2)_K$.

For $(2)_K$, note that $k \geq K \geq |A_0|$ implies that $K \times c = \pi \times c = k \times c$, for all $c \in A_0$. Consequently,

$$a(K \times x)(c) = a(k \times x)(c) \quad \text{and, therefore,} \quad (a(K \times x))^\pi(c) = (a(k \times x))^\pi(c).$$

This implies the claim.

For $(1)_K$, fix $a \in A_n$ and $c \in A_0$. If $n \leq K - m_0$, then $K - n \geq m_0 = |A_0|$ implies that $(K - n) \times c = \pi \times c$. Consequently,

$$a(c, \dots, c) + (K - n) \times c = a(c, \dots, c) + \pi \times c = a(c, \dots, c) + \pi \times c + c$$

and we are done. Thus, we may assume that $n > K - m_0 = m_0^{2m_1}$. As \mathfrak{A} is generated by $A_0 \cup A_1$, there exists some forest $s \in \mathbb{F}_i(A_0 \cup A_1)$ with $\pi(s) = a$. We distinguish several cases.

If some of the variables x_0, \dots, x_{n-1} does not appear in s , we can use $(1)_k$ to show that

$$\begin{aligned} a(c, \dots, c, \dots, c) + (K - n) \times c &= a(c, \dots, c + \dots + c, \dots, c) + (K - n) \times c \\ &= a(c, \dots, k \times c, \dots, c) + (K - n) \times c \\ &= a(c, \dots, k \times c, \dots, c) + (K - n) \times c + c. \end{aligned}$$

Next, suppose that s is highly branching in the sense that it has the form

$$s = r(t_0 + \dots + t_{m_0^2-1})$$

where each subterm t_i contains some variable. Then there are indices $i_0 < \dots < i_{m_0-1}$ such that $\pi(t_{i_0}(\bar{c})) = \dots = \pi(t_{i_{m_0-1}}(\bar{c}))$ (where \bar{c} denotes as many copies of c as appear in the respective term). Hence, $(1)_k$ again implies that

$$\begin{aligned} a(\bar{c}) + (K - n) \times c &= \pi(s(\bar{c})) + (K - n) \times c \\ &= \pi(r(t_0(\bar{c}) + \dots + t_{m_0^2-1}(\bar{c}))) + (K - n) \times c \\ &= \pi(r(t_0(\bar{c}) + \dots + t_{m_0^2-1}(\bar{c}) + k \times t_{i_0}(\bar{c}))) + (K - n) \times c \\ &= a(\bar{c}) + (K - n) \times c + c. \end{aligned}$$

Note that a tree of height $h := m_1$ where every vertex has at most $d := m_0^2$ successors has at most $d^h = m_0^{2m_1}$ leaves. Hence, if s is not highly branching in the sense above, the fact that it contains $n > m_0^{2m_1}$ variables implies that there must be a chain $v_0 \prec \dots \prec v_{m_1}$ of vertices such that, for every $i < m_1$, there is some leaf u labelled by a variable with $v_{i-1} \prec u$ and $v_i \not\prec u$. (For $i = 0$, we omit the first condition.) Hence, we can decompose s as

$$s(\bar{c}) = r_0(\bar{c}, r_1(\bar{c}, \dots, r_{m_1}(\bar{c}))),$$

and there are two indices $i < j$ such that

$$\pi(r_0(\bar{c}, \dots, r_i(\bar{c}, x))) = \pi(r_0(\bar{c}, \dots, r_j(\bar{c}, x))).$$

Consequently, we can use pumping to obtain a term

$$\pi(s(\bar{c})) = \pi(r_0(\bar{c}, \dots, r_i(\bar{c}, x)) [r_{i+1}(\bar{c}, \dots, r_j(\bar{c}, x))]^k r_{j+1}(\bar{c}, \dots, r_{m_1}(\bar{c})))$$

which contains at least k occurrences of c , and the claim follows again by $(1)_k$. \blacktriangleleft

According to this lemma, we can check for cEF-definability of a language L , by computing its syntactic algebra $\mathfrak{S}(L)$, the associated constant K , and then checking the equations for $k = K$.

► **Corollary 4.5.** *It is decidable whether a given regular language L is cEF-definable.*

When taking the special case of $k = 1$ in Theorem 4.2, we obtain the following characterisation of EF-definability.

► **Theorem 4.6.** *A forest language $L \subseteq \mathbb{F}_0\Sigma$ is definable in the logic EF if, and only if, the syntactic algebra $\mathfrak{S}(L)$ satisfies the following equations:*

$$\begin{array}{ll}
 c + d = d + c & (a(x) + b(x))^\omega = (ab(x))^\omega \\
 (ab)^\pi = b(ab)^\pi & (a(x) + c)^\omega = (a(x + c))^\omega \\
 (abb')^\omega = (ab'b)^\omega & (a(x + c + c))^\omega = (a(x + c))^\omega \\
 (aab)^\omega = (ab)^\omega & [a(b(x_0, x_1))^{\omega_1}]^{\omega_0} = [ab(x_0, x_0)]^{\omega_0} \\
 ac = ac + c \quad c = c + c & [a(x + a^\pi c)]^\omega = a^\pi c,
 \end{array}$$

for all $a, b, b' \in S_1(L)$ and $c, d \in S_0(L)$.

► **Corollary 4.7.** *It is decidable whether a given regular language L is EF-definable.*

References

- 1 A. Blumensath. Branch-Continuous Tree Algebras. unpublished. [arXiv:1807.04568](#).
- 2 A. Blumensath. Recognisability for algebras of infinite trees. *Theoretical Computer Science*, 412:3463–3486, 2011.
- 3 A. Blumensath. An Algebraic Proof of Rabin’s Tree Theorem. *Theoretical Computer Science*, 478:1–21, 2013.
- 4 A. Blumensath. Regular Tree Algebras. *Logical Methods in Computer Science*, 16:16:1–16:25, 2020.
- 5 A. Blumensath. Algebraic Language Theory for Eilenberg–Moore Algebras. *Logical Methods in Computer Science*, 17:6:1–6:60, 2021.
- 6 M. Bojańczyk. Recognisable languages over monads. unpublished note. [arXiv:1502.04898v1](#).
- 7 M. Bojańczyk and T. Idziaszek. Algebra for Infinite Forests with an Application to the Temporal Logic EF. In *Proc. 20th International Conference on Concurrency Theory, CONCUR, LNCS 5710*, pages 131–145, 2009.
- 8 M. Bojańczyk, T. Idziaszek, and M. Skrzypczak. Regular languages of thin trees. In *Proc. 30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013*, pages 562–573, 2013.
- 9 M. Bojańczyk and B. Klin. A non-regular language of infinite trees that is recognizable by a finite algebra. *Logical Methods in Computer Science*, 15, 2019.
- 10 M. Bojańczyk and I. Walukiewicz. Forest Algebras. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata: History and Perspectives*, pages 107–132. Amsterdam University Press, 2007.
- 11 T. Colcombet and A. Jaquard. A Complexity Approach to Tree Algebras: the Bounded Case. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12–16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 127:1–127:13, 2021.

A

 The proof of Theorem 4.2

For the proof of Theorem 4.2, we need to set up a bit of machinery. We start by defining the suitable notion of bisimulation for cEF_k . The difference to the standard notion is that we use reachability instead of the edge relation and that we also have to preserve the number of reachable positions.

► **Definition A.1.** Let $m, k < \omega$.

(a) For trees $s, t \in \mathbb{F}\Sigma$, we define

$$\begin{aligned}
 s \approx_k^0 t & \quad \text{: iff} \quad \text{the roots of } s \text{ and } t \text{ have the same label} \\
 s \approx_k^{m+1} t & \quad \text{: iff} \quad \text{the roots of } s \text{ and } t \text{ have the same label,} \\
 & \quad \text{for every } k\text{-tuple } \bar{x} \text{ in } \text{dom}(s) \text{ not containing the root, there is} \\
 & \quad \text{some } k\text{-tuple } \bar{y} \text{ in } \text{dom}(t) \text{ not containing the root such that} \\
 & \quad s|_{x_i} \approx_k^m t|_{y_i} \quad \text{for all } i < k \text{ and,} \\
 & \quad \text{for every } k\text{-tuple } \bar{y} \text{ in } \text{dom}(t) \text{ not containing the root, there is} \\
 & \quad \text{some } k\text{-tuple } \bar{x} \text{ in } \text{dom}(s) \text{ not containing the root such that} \\
 & \quad s|_{x_i} \approx_k^m t|_{y_i} \quad \text{for all } i < k.
 \end{aligned}$$

To simplify notation, we will frequently write $x \approx_k^m y$ for vertices x and y instead of the more cumbersome $s|_x \approx_k^m t|_y$.

(b) For forests $s, t \in \mathbb{F}\Sigma$ with possibly several components, we set

$$\begin{aligned}
 s \sim_k^{m+1} t & \quad \text{: iff} \quad \text{for every } k\text{-tuple } \bar{x} \text{ in } s \text{ there is some } k\text{-tuple } \bar{y} \text{ in } t \text{ such that} \\
 & \quad s|_{x_i} \approx_k^m t|_{y_i} \quad \text{for all } i < k \text{ and,} \\
 & \quad \text{for every } k\text{-tuple } \bar{y} \text{ in } t \text{ there is some } k\text{-tuple } \bar{x} \text{ in } s \text{ such that} \\
 & \quad s|_{x_i} \approx_k^m t|_{y_i} \quad \text{for all } i < k. \quad \lrcorner
 \end{aligned}$$

Let us show that this notion of bisimulation captures the expressive power of cEF . The proof is mostly standard. We start by introducing the following notion of a type.

► **Definition A.2.**

(a) We define the *type* $\text{tp}_k^m(s)$ of a tree $s \in \mathbb{F}\Sigma$ by

$$\text{tp}_k^0(s) := s(\langle \rangle) \quad \text{and} \quad \text{tp}_k^{m+1}(s) := \langle s(\langle \rangle), \theta_s \rangle$$

where $\langle \rangle$ denotes the root of s and

$$\begin{aligned}
 \theta_s := \{ \langle l, \sigma \rangle \mid l \leq k, x_0, \dots, x_{l-1} \in \text{dom}(s) \text{ distinct, not equal to the root,} \\
 \sigma = \text{tp}_k^m(s|_{x_0}) = \dots = \text{tp}_k^m(s|_{x_{l-1}}) \}.
 \end{aligned}$$

(b) For an arbitrary forest $s \in \mathbb{F}\Sigma$, we set $\text{Tp}_k^{m+1}(s) := \theta_s$, where

$$\begin{aligned}
 \theta_s := \{ \langle l, \sigma \rangle \mid l \leq k, x_0, \dots, x_{l-1} \in \text{dom}(s) \text{ distinct,} \\
 \sigma = \text{tp}_k^m(s|_{x_0}) = \dots = \text{tp}_k^m(s|_{x_{l-1}}) \}. \quad \lrcorner
 \end{aligned}$$

As standard proof establishes the following equivalences.

► **Lemma A.3.** *Let $k, m < \omega$.*

(a) *For trees $s, t \in \mathbb{F}_0\Sigma$, the following statements are equivalent.*

- (1) $s \approx_k^m t$
- (2) $\text{tp}_k^m(s) = \text{tp}_k^m(t)$
- (3) $s \models \varphi \Leftrightarrow t \models \varphi$, for all $\varphi \in \text{cEF}_k^m$.

(b) *For arbitrary forests $s, t \in \mathbb{F}_0\Sigma$, the following statements are equivalent.*

- (1) $s \sim_k^m t$
- (2) $\text{Tp}_k^m(s) = \text{Tp}_k^m(t)$
- (3) $s \models \varphi \Leftrightarrow t \models \varphi$, for all $\varphi \in \text{cEF}_k^m$.

► **Corollary A.4.** *A language $L \subseteq \mathbb{F}\Sigma$ is cEF_k^m -definable if, and only if, it is regular and satisfies*

$$s \sim_k^m t \text{ implies } s \in L \Leftrightarrow t \in L, \text{ for all regular forests } s, t \in \mathbb{F}_0\Sigma.$$

Proof. (\Rightarrow) follows by the implication (1) \Rightarrow (3) of Lemma A.3.

(\Leftarrow) Set

$$\varphi := \bigvee \{ \chi_\tau \mid \tau = \text{Tp}_k^m(s) \text{ for some regular forest } s \in L \},$$

where χ_τ are the formulae from the proof of Lemma A.3. For a regular forest $t \in \mathbb{F}_0\Sigma$, it follows that

$$\begin{aligned} t \models \varphi & \text{ iff } \text{Tp}_k^m(t) = \text{Tp}_k^m(s), \text{ for some regular forest } s \in L, \\ & \text{ iff } t \sim_k^m s, \text{ for some regular forest } s \in L, \\ & \text{ iff } t \in L. \end{aligned}$$

Let K be the language defined by φ . Since L and K are both regular languages that contain the same regular forests, it follows that $L = K$. Thus, L is cEF_k^m -definable. ◀

We want to show that an algebra recognises cEF_k -definable languages if, and only if, it satisfies the following equations.

► **Definition A.5.**

(a) A forest algebra \mathfrak{A} is an *algebra for cEF_k* if it is finitary, generated by $A_0 \cup A_1$, and satisfies the following equations.

$$\text{(G1)}_k \quad a_n(c, \dots, c) + (k - n) \times c = a_n(c, \dots, c) + (k - n + 1) \times c$$

$$\text{(G1)} \quad (ab)^\pi = b(ab)^\pi$$

$$\text{(G2)} \quad a^\omega + a^\omega = a^\omega$$

$$\text{(G3)} \quad c + d = d + c$$

$$\text{(G4)} \quad (a(x) + b(x))^\omega = (ab(x))^\omega$$

$$\text{(G5)} \quad (a(x) + c)^\omega = (a(x + c))^\omega$$

$$\text{(G6)} \quad (a(x + c + c))^\omega = (a(x + c))^\omega$$

$$\text{(G7)} \quad [a(b(x_0, x_1))^\omega]^\omega = [ab(x_0, x_0)]^\omega$$

$$\text{(G8)} \quad (abb')^\omega = (ab'b)^\omega$$

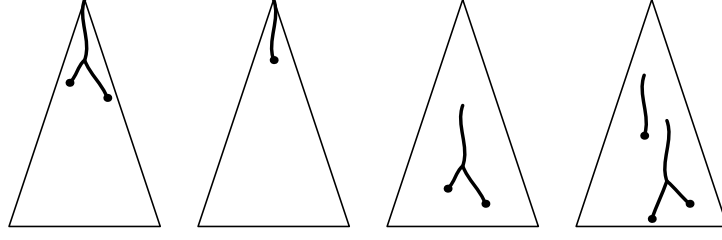
$$\text{(G9)} \quad (aab)^\omega = (ab)^\omega$$

$$\text{(G10)} \quad [a(x + bc + c)]^\omega = [a(x + bc)]^\omega$$

$$\text{(G12)}_k \quad [a(x + (a(k \times x))^\pi(c))]^\omega = k \times (a(k \times x))^\pi(c)$$

where $a, b, b' \in A_1$, $c, d \in A_0$, $a_n \in A_n$, and $n \leq k$.

(b) A forest algebra \mathfrak{A} is an *algebra for cEF* if it is an algebra for cEF_k , for some $k \geq 1$. ◻



■ **Figure 3** A forest s with a convex set U (in bold) that has three close U -ends (on the left) and five far ones (on the right). The height is $h(s, U) = 2$.

In the proof that algebras for cEF recognise exactly the cEF-definable languages, we use one of the Green's relations (suitably modified for forest algebras).

► **Definition A.6.** Let \mathfrak{A} be a forest algebra. For $a, b \in A_0$, we define

$$a \leq_L b \quad \text{iff} \quad a = c(b) \quad \text{or} \quad a = b + d, \quad \text{for some } c \in A_1, d \in A_0. \quad \lrcorner$$

► **Lemma A.7.** Let \mathfrak{A} be an algebra for cEF_k .

- (a) The relation \leq_L is antisymmetric.
 (b) For $a \in A_1$, $c \in A_0$, we have

$$\begin{aligned} c = c + c & \quad \text{implies} \quad ac = ac + c, \\ c = a(c, c) & \quad \text{implies} \quad c = c + c. \end{aligned}$$

Proof.

- (a) For a contradiction, suppose that there are elements $a \neq b$ with $a \leq_L b \leq_L a$. By definition, we can find elements c and d such that (1) $a = c(b)$ or (2) $a = b + c$, and (i) $b = d(a)$ or (ii) $b = a + d$. We have thus to consider four cases. In each of them we obtain a contradiction via $(G1)_k$ or $(G2)$.

$$\begin{aligned} (1, i) \quad a = cb = cda = (cd)^\pi(a) = d(cd)^\pi(a) = da = b. \\ (1, ii) \quad a = cb = c(a + d) = (c(x + d))^\pi(a) = (c(x + d))^\pi(a) + d = a + d = b. \\ (2, i) \quad b = da = d(b + c) = (d(x + c))^\pi(b) = (d(x + c))^\pi(b) + c = b + c = a. \\ (2, ii) \quad a = b + c = a + d + c = a + k \times (d + c) = a + k \times (d + c) + d = a + d = b. \end{aligned}$$

- (b) By $(G1)_k$ we have

$$\begin{aligned} c = c + c & \quad \text{implies} \quad ac = a(c + c) = a(k \times c) = a(k \times c) + c = ac + c, \\ c = a(c, c) & \quad \text{implies} \quad c = a(c, c) = (a(x, c))^\pi(c) = (a(x, c))^\pi(c) + c = c + c. \quad \blacktriangleleft \end{aligned}$$

Let us take a look at the following situation (see Figure 3). Let s be a forest and U a set of vertices. We assume that U is *convex* in the sense that $u \preceq v \preceq w$ and $u, w \in U$ implies $v \in U$ (where \preceq denotes the forest order). We call the maximal elements (w.r.t. \preceq) of U the *U-ends*. An U -end u is *close* if $u' \in U$, for all $u' \preceq u$. Otherwise, it is *far*. We would like to know how many of the U -ends are close.

► **Lemma A.8.** Let $m \geq 0$ and $k \geq 1$, let $s \sim_k^{m+k+2} t$ be two forests, $U \subseteq \text{dom}(s)$ a convex set that is closed under \approx_k^m , and set

$$V := \{v \in \text{dom}(t) \mid u \approx_k^m v \text{ for some } u \in U\}.$$

- (a) V is convex and closed under \approx_k^m .
- (b) The numbers of ends of U and V are the same, or both numbers are at least k .
- (c) If U has less than k ends, then U is finite if, and only if, V is finite.
- (d) If U is finite and has less than k ends, then U and V have the same numbers of close ends and of far ends.

Proof.

- (a) If V is not convex, there are vertices $v \prec v' \prec v''$ of t with $v, v'' \in V$ and $v' \notin V$. Fix vertices $u \prec u' \prec u''$ with $u \approx_k^{m+2} v$, $u' \approx_k^{m+1} v'$, and $u'' \approx_k^m v''$. By definition of V , we have $u, u'' \in U$ and $u' \notin U$. This contradicts the fact that U is convex.
To see that V is closed under \approx_k^m , suppose that $v \in V$ and $v \approx_k^m v'$. By definition of V , there is some $u \in U$ with $u \approx_k^m v$. Hence, $u \approx_k^m v \approx_k^m v'$. As \approx_k^m is transitive, this implies that $v' \in V$.
- (b) For a contradiction, suppose that U has $n < k$ ends while V has more than n ends. (The other case follows by symmetry.) Choose $n + 1$ ends $v_0, \dots, v_n \in V$. Since $s \approx_k^{m+2} t$, there are vertices u_0, \dots, u_n in s with $u_i \approx_k^{m+1} v_i$. By definition of V , we have $u_i \in U$. By assumption, there is some index j such that u_j is not an end. Hence, we can find a vertex $u' \succ u_j$ with $u' \in U$. Fix a vertex $v' \succ v_j$ of t with $u' \approx_k^m v'$. Then $v' \in V$ and v_j is not an end. A contradiction.
- (c) For a contradiction, suppose that U is finite, but V is not. (The other case follows by symmetry.) By (b), V has only finitely many ends. Hence, there is some element $v \in V$ such that $v \not\leq v'$ for every end v' of V . Since $s \approx_k^{m+3} t$, we can find a vertex u of s with $u \approx_k^{m+2} v$. This implies that $u \in U$. As U is finite, we can find some end u' of U with $u \leq u'$. Fix some $v' \geq v$ with $u' \approx_k^{m+1} v'$. Then $u' \in U$ implies $v' \in V$. By choice of v , there is some $v'' \succ v'$ with $v'' \in V$. Choose $u'' \succ u'$ with $u'' \approx_k^m v''$. By choice of u' , we have $u'' \notin U$. This contradicts the fact that $v'' \in V$.
- (d) By (b), we only need to prove that the number of close ends is the same. Let \hat{U} and \hat{V} be the sets of U -ends and V -ends, respectively. We denote by $N(s, U)$ the number of close U -ends and by $F(s, U)$ the set of all proper subforests s' of s that are attached to some vertex v that does not belong to U but where at least one root belongs to U . (A forest s' is a *proper subforest* of s attached at v if s' can be obtained from the subtree $s|_v$ by removing the root v .) We define the following equivalence relation.

$$\begin{aligned} \langle s, U \rangle \asymp_0 \langle t, V \rangle & \quad \text{: iff } N(s, U) = N(t, V), \\ \langle s, U \rangle \asymp_{i+1} \langle t, V \rangle & \quad \text{: iff } N(s, U) = N(t, V) \text{ and} \\ & \quad \#_\tau(s, U) = \#_\tau(t, V), \text{ for every } \asymp_i\text{-class } \tau, \end{aligned}$$

where $\#_\tau(s, U)$ denotes the number of subforests $s' \in F(s, U)$ that belong to the class τ . We define the U -height of s by

$$h(s, U) := \begin{cases} 0 & \text{if } F(s, U) = \emptyset \\ 1 + \max \{ h(s', U) \mid s' \in F(s, U) \} & \text{otherwise.} \end{cases}$$

By induction on l , we will prove the following claim:

$$(*) \quad s \sim_k^{m+l+2} t \quad \text{and} \quad h(s, U) \leq l \quad \text{implies} \quad h(s, U) = h(t, V) \quad \text{and} \quad \langle s, U \rangle \asymp_l \langle t, V \rangle.$$

As $h(s, U) \leq |\hat{U}| < k$, it then follows that $\langle s, U \rangle \asymp_k \langle t, V \rangle$. In particular, $N(s, U) = N(t, V)$, as desired.

It thus remains to prove (*). First, consider the case where $l = 0$. If $h(t, V) > 0$, there is some V -end v that is not close. Fix some vertex $v' \prec v$ with $v' \notin V$. Since $s \sim_k^{m+2} t$, we can find vertices $u' \prec u$ of s with $u' \approx_k^{m+1} v'$ and $u \approx_k^m v$. By definition of V , it follows that $u' \notin U$ and $u \in U$. As U is finite, we can find some U -end $w \succeq u$. But $u' \prec u \preceq w$ implies that w is not close. Hence, $h(s, U) > 0$. A contradiction.

For the second part, suppose that $\langle s, U \rangle \not\approx_0 \langle t, V \rangle$, that is, $N(s, U) \neq N(t, V)$. By symmetry, we may assume that $m := N(s, U) < N(t, v)$. Pick $m + 1$ distinct close V -ends v_0, \dots, v_m . Since $m + 1 \leq k$ and $s \sim_k^{m+2} t$, there are elements $u_0, \dots, u_m \in \text{dom}(s)$ with $u_i \approx_k^{m+1} v_i$. There must be some index j such that u_j is not a close U -end. As U is closed under \approx_k^m and $u_j \approx_k^m v_j \approx_k^m u$, for some $u \in U$, it follows that $u_j \in U$. Furthermore, $u_j \approx_k^{m+1} v_j$ and the fact that v_j is a V -end implies that $u' \notin U$, for all $u' \succ u_j$. Thus, u_j is a U -end. But $h(s, U) = 0$ implies that all U -ends of s are close. A contradiction.

For the inductive step, suppose that $s \sim_k^{m+(l+1)+2} t$ holds but we have $h(s, U) \neq h(t, V)$ or $\langle s, U \rangle \not\approx_{l+1} \langle t, V \rangle$. We distinguish several cases.

- (i) Suppose that $h(s, U) > h(t, V)$. By definition of h , there is a subforest $s' \in F(s, U)$ with $h(s', U) = h(s, U) - 1$. Then there is some subforest t' of t with $s' \sim_k^{m+l+2} t'$. By inductive hypothesis it follows that

$$h(s, U) = h(s', U) + 1 = h(t', V) + 1 < h(t, V) + 1 \leq h(s, U).$$

A contradiction.

- (ii) Suppose that $h(s, U) < h(t, V)$. By definition of h , there is a subforest $t' \in F(t, V)$ with $h(t', V) = h(t, V) - 1$. Fix a subforest s' of s with $s' \sim_k^{m+l+2} t'$. By inductive hypothesis, it follows that

$$h(s, U) > h(s', U) = h(t', V) = h(t, V) - 1 \geq h(s, U).$$

A contradiction.

- (iii) Suppose that $N(s, U) \neq N(t, v)$ and there is no \approx_l -class τ with $\#_\tau(s, U) \neq \#_\tau(t, V)$. Then we have $|\hat{U}| - N(s, U) = |\hat{V}| - N(t, V)$. Since $|\hat{U}| = |\hat{V}|$ it follows that $N(s, U) = N(t, V)$. A contradiction.

- (iv) Finally, suppose that there is some \approx_l -class τ with $\#_\tau(s, U) \neq \#_\tau(t, V)$. By symmetry, we may assume that $m := \#_\tau(s, U) < \#_\tau(t, V)$. We choose $m + 1$ vertices v_0, \dots, v_m of t such that the attached subforests have class τ . Since $s \sim_k^{m+(l+1)+2} t$ and $m + 1 \leq k$, there are vertices u_0, \dots, u_m of s such that $u_i \sim_k^{m+l+2} v_i$, for all $i \leq m$. Let s_i be the subforest of s attached to u_i , and t_i the subforest of t attached to v_i . By inductive hypothesis, it follows that $s_i \approx_l t_i$, for $i \leq m$. Thus, s has at least $m + 1$ different subforest in the class τ . A contradiction. \blacktriangleleft

► **Proposition A.9.** *Let \mathfrak{A} be an algebra for cEF_k . Then*

$$s \approx_k^{(k+3)(|A_0|+1)} t \text{ implies } \pi(s) = \pi(t), \text{ for all regular trees } s, t \in \mathbb{F}_0(A_0 \cup A_1).$$

Proof. Let m be the number of L -classes above $b := \pi(s)$ (including that of b itself). We will prove by induction on m that

$$s \approx_k^{f(m)} t \text{ implies } \pi(t) = b,$$

where $f(m) := (m + 1)(k + 3)$. Set

$$S := \{x \in \text{dom}(s) \mid \pi(s|_x) = b\},$$

$$T := \{y \in \text{dom}(t) \mid x \approx^{f(m-1)} y \text{ for some } x \in S\}.$$

As t is regular it is the unravelling of some finite graph G . For each $y \in T$, we will prove that $\pi(t|_y) = b$ by induction on the number of strongly connected components of G that are contained in T and that are reachable from y . Hence, fix $y \in T$, let C be the strongly connected component of G containing y , and choose some $x \in S$ with $x \approx_k^{f(m)-1} y$. We distinguish two cases.

(a) Let us begin our induction with the case where C is trivial, i.e., it consists of the single vertex y without self-loop. Then

$$t|_y = a(t_0 + \cdots + t_{n-1} + t'_0 + \cdots + t'_{q-1})$$

where $a := t(y)$ and the subtrees t_i lie outside of T while the t'_i contain vertices in T . Set $d_i := \pi(t_i)$. By our two inductive hypotheses, we already know that $\pi(t'_i) = b$ and that $b <_{\mathbb{L}} d_i$. Hence,

$$\pi(t|_y) = a(d_0 + \cdots + d_{n-1} + q \times b).$$

We have to show that this value is equal to b . Suppose that

$$s|x = a(s_0 + \cdots + s_{l-1} + s'_0 + \cdots + s'_{p-1}),$$

where again the trees s_i lie outside of S , while the s'_i contain vertices of S . Setting $c_i := \pi(s_i)$ it follows that

$$\pi(s|x) = a(c_0 + \cdots + c_{l-1} + p \times b).$$

Since $x \in S$, we already know that this value is equal to b . Hence, it remains to show that

$$a(c_0 + \cdots + c_{l-1} + p \times b) = a(d_0 + \cdots + d_{n-1} + q \times b).$$

We start by proving that

$$c_0 + \cdots + c_{l-1} = d_0 + \cdots + d_{n-1}.$$

By (G4) it is sufficient to prove that, for every $c \in A_0$, the number of occurrences of the value c in the sum on the left-hand side is either the same as that on the right-hand side, or that we can add an arbitrary number of c on both sides without changing the respective values. Hence, consider some element $c \in A_0$ where these numbers are different. Let U be the set of all vertices $u \succ x$ such that $\pi(s|_u) = c$ and let V be the set of vertices $v \succ y$ with $\pi(t|_v) = c$. As $\leq_{\mathbb{L}}$ is antisymmetric, these two sets are convex. Furthermore, by inductive hypothesis on m , they are also closed under $\approx_k^{f(m-1)}$. Since $f(m) - 1 = f(m-1) + k + 2$, we can therefore apply Lemma A.8 and we obtain one of the following cases.

(i) U and V both have at least k ends. Then we can write $s_0 + \cdots + s_{l-1}$ as $r(s'_0, \dots, s'_{k-1})$ with $\pi(s'_i) = c$. Hence, it follows by (G1) $_k$ that

$$\begin{aligned} c_0 + \cdots + c_{l-1} &= \pi(r)(c, \dots, c) = \pi(r)(c, \dots, c) + \pi \times c \\ &= c_0 + \cdots + c_{l-1} + \pi \times c. \end{aligned}$$

For t it follows in the same way that

$$d_0 + \cdots + d_{n-1} = d_0 + \cdots + d_{n-1} + \pi \times c.$$

Consequently, we can add an arbitrary number of terms c to both sides of the above equation and thereby make their numbers equal.

- (ii) Both U and V are infinite, but each has less than k ends. Thus, U contains an infinite path and we can use Ramsey's Theorem (or the fact that s is regular) to write $\pi(s_0 + \cdots + s_{l-1})$ as $a'e^\omega$ where $ec = c = e^\omega$. By (G3) and (G1) $_k$ it follows that

$$\begin{aligned} c_0 + \cdots + c_{l-1} &= a'e^\omega = a'(e^\omega + \cdots + e^\omega) = a'(c + \cdots + c) \\ &= a'(c + \cdots + c) + \pi \times c \\ &= c_0 + \cdots + c_{l-1} + \pi \times c. \end{aligned}$$

For $t|_y$, we similarly obtain

$$d_0 + \cdots + d_{n-1} = d_0 + \cdots + d_{n-1} + \pi \times c,$$

and we can equalise the number of c as in Case (i).

- (iii) The last remaining case is where both U and V are finite and they have the same number of close ends. Then the sums $c_0 + \cdots + c_{l-1}$ and $d_0 + \cdots + d_{n-1}$ contain the same number of terms with value c and there is nothing to prove.

We have thus shown that

$$c_0 + \cdots + c_{l-1} = d_0 + \cdots + d_{n-1}.$$

If $p = q$, we are done. Hence, we may assume that $p \neq q$. To conclude the proof, we set

$$U := \{u \in S \mid x \prec u\} \quad \text{and} \quad V := \{v \in T \mid y \prec v\}.$$

If $p > 0$, then $x \approx_k^{f(m)-1} y$ and $U \neq \emptyset$ implies $V \neq \emptyset$. Hence, $q > 0$. In the same way, $q > 0$ implies $p > 0$. Consequently, we have $p, q > 0$. We consider several cases.

- (i) If $b + b = b$, then

$$\begin{aligned} a(d_0 + \cdots + d_{n-1} + q \times b) &= a(c_0 + \cdots + c_{l-1} + q \times b) \\ &= a(c_0 + \cdots + c_{l-1} + p \times b) = b, \end{aligned}$$

as desired.

- (ii) If U is not a chain, we obtain $b = a'(b, b)$, for some a' , and Lemma A.7 implies that we are in Case (i).
- (iii) If U contains an infinite chain, we can use Ramsey's Theorem (or the fact that s is regular), to obtain a factorisation $b = e^\omega$, which implies that $b + b = b$ by (G3). Hence, we are in Case (i) again.
- (iv) If U is a finite chain, then so is V , by Lemma A.8. Hence, $p = 1 = q$ and we are done.

- (b) It remains to consider the case where C is not trivial. Then we can factorise

$$t|_y = r(t_0, \dots, t_{n-1}, t'_0, \dots, t'_{q-1}),$$

where $r \in \mathbb{F}A$ is the unravelling of C , the subtrees t_i lie outside of T , while the subtrees t'_i contain vertices in T . Setting $d_i := \pi(t_i)$, it follows by the two inductive hypotheses that $d_i >_{\perp} b$ and $\pi(t'_i) = b$. Consequently,

$$\pi(t|_y) = \pi(r)(d_0, \dots, d_{n-1}, b, \dots, b).$$

Let us simplify the term r . Introducing one variable x_v , for every vertex $v \in C$, we can write r as a system of equations

$$x_v = a_v(x_{u_0} + \cdots + x_{u_{l-1}} + c_0 + \cdots + c_{q-1}), \quad \text{for } v \in C,$$

where u_0, \dots, u_{l-1} are the successors of v that belong to C and c_0, \dots, c_{q-1} are constants from $\{d_0, \dots, d_{n-1}, b\}$ that correspond to successors outside of C . Solving this system of equations, we obtain a finite term r_0 built up from elements of $A_0 \cup A_1$ using as operations the horizontal product, the vertical product, and the ω -power operation, such that

$$\pi(t|_y) = \pi(r_0)(d_0, \dots, d_{n-1}, b).$$

With the help of the equations (G5)–(G10), we can transform r_0 in several steps (while preserving its product) until it assumes the form

$$\begin{aligned} & [a_0 \cdots a_{j-1}(x + d_0 + \cdots + d_{n-1} + b)]^\omega \\ \text{or } & [a_0 \cdots a_{j-1}(x + d_0 + \cdots + d_{n-1})]^\omega \end{aligned}$$

where a_0, \dots, a_{j-1} are the labels of the vertices in C .

We distinguish two cases. First suppose that there is no term with value b in the above sum. This means that every subtree attached to C lies entirely outside of the set T . Then $x \approx_k^{f(m)-1} y$ implies that we can factorise $s|_x$ as

$$s|_x = r'(s_0, \dots, s_{l-1})$$

where

- $\{\pi(s_0), \dots, \pi(s_{l-1})\} = \{d_0, \dots, d_{n-1}\}$,
- all labels of r' are among a_0, \dots, a_{j-1} ,
- every vertex of r' has, for every $i < k$, some descendant labelled a_i .

As above we can transform $s|_x$ into

$$[a_0 \cdots a_{j-1}(x + c_0 + \cdots + c_{l-1})]^\omega$$

where $c_i := \pi(s_i)$. Since $\{c_0, \dots, c_{l-1}\} = \{d_0, \dots, d_{n-1}\}$ it follows that

$$\begin{aligned} \pi(t|_y) &= (a_0 \cdots a_{j-1}(x + d_0 + \cdots + d_{n-1}))^\omega \\ &= (a_0 \cdots a_{j-1}(x + c_0 + \cdots + c_{l-1}))^\omega = \pi(s|_x) = b. \end{aligned}$$

It thus remains to consider the case where some term has value b . Using (G7) and (G11) and the fact that $b <_{\mathbb{L}} d_i$, it then follows that

$$\pi(t|_y) = [a_0 \cdots a_{j-1}(x + d_0 + \cdots + d_{n-1} + b)]^\omega = [a_0 \cdots a_{j-1}(x + b)]^\omega.$$

For every $i < j$, we fix some $z_i \in S$ with label a_i such that $x \prec z_i$ and some successor of z_i also belongs to S . Then

$$\pi(s|_{z_i}) = a_i(c_0^i + \cdots + c_{i-1}^i + b + \cdots + b),$$

for some $c_0^i, \dots, c_{i-1}^i >_{\mathbb{L}} b$. Since

$$b = \pi(s|_{z_i}) = a_i(c_0^i + \cdots + c_{i-1}^i + b + \cdots + b) \leq_{\mathbb{L}} c_0^i + \cdots + c_{i-1}^i + b + \cdots + b \leq_{\mathbb{L}} b$$

it follows by asymmetry of \leq_L that

$$c_0^i + \cdots + c_{i+1}^i + b + \cdots + b = b \quad \text{and} \quad a_i(b) = a_i(c_0^i + \cdots + c_{i+1}^i + b + \cdots + b) = b.$$

Consequently, $a_0 \cdots a_{j-1} b = b$, which implies that $a^\pi b = b$ where $a := a_0 \cdots a_{j-1}$. We claim that $b + b = b$. It then follows that

$$b = a(b) = a(k \times x)(b) = (a(k \times x))^\pi(b),$$

which, by $(G12)_k$, implies that

$$\pi(t|_y) = [a(x + b)]^\omega = [a(x + a(k \times x)^\pi(b))]^\omega = k \times a(k \times x)^\pi(b) = k \times b = b,$$

as desired.

Hence, it remains to prove our claim that $b + b = b$. By our assumption on y and C , there is some vertex $u \in C$ that has some successor $v \notin C$ with $v \in T$. Since $s|_x \approx_k^{f(m)-1} t|_y$ and $f(m) \geq f(m-1) + k + 1$, there are vertices $x \preceq u_0 < \cdots < u_{k-1}$ each of which has some successor $v_i \in S$ with $v_i \not\preceq u_{i+1}$. Consequently, we can write

$$\pi(s|_x) = a' a''(b, \dots, b) \quad \text{and} \quad \pi(s|_{u_0}) = a''(b, \dots, b),$$

where $a' \in A_1$ and $a'' \in A_k$. Hence, it follows by $(G1)_k$ that

$$b + b = \pi(s|_{u_0}) + b = a''(b, \dots, b) + b = a''(b, \dots, b) = \pi(s|_{u_0}) = b. \quad \blacktriangleleft$$

► **Theorem A.10.** *A regular forest algebra \mathfrak{A} is an algebra for cEF_k if, and only if, there exists a number $m < \omega$ such that*

$$s \sim_k^m t \quad \text{implies} \quad \pi(s) = \pi(t), \quad \text{for all regular forests } s, t \in \mathbb{F}(A_0 \cup A_1).$$

Proof. (\Leftarrow) In each of the equations $(G1)_k$ – $(G12)_k$, the two terms on both sides are \sim_k^m -equivalent.

(\Rightarrow) By Proposition A.9, there is some number m such that

$$s \approx_k^m t \quad \text{implies} \quad \pi(s) = \pi(t), \quad \text{for regular trees } s, t \in \mathbb{F}(A_0 \cup A_1).$$

Let $s, t \in \mathbb{F}(A_0 \cup A_1)$ be regular forests. We claim that

$$s \sim_k^{m+k+2} t \quad \text{implies} \quad \pi(s) = \pi(t).$$

Suppose that $s = s_0 + \cdots + s_{l-1}$ and $t = t_0 + \cdots + t_{n-1}$, for trees s_i and t_i , and set $c_i := \pi(s_i)$ and $d_i := \pi(t_i)$. Analogous to Part (a) of the proof of Proposition A.9, we can use Lemma A.8 to show that

$$\pi(s) = c_0 + \cdots + c_{l-1} = d_0 + \cdots + d_{n-1} = \pi(t). \quad \blacktriangleleft$$

We complete the proof of Theorem 4.2 as follows.

► **Theorem A.11.** *A regular language $L \subseteq \mathbb{F}_0 \Sigma$ is cEF_k -definable if, and only if, its syntactic algebra $\mathfrak{S}(L)$ is an algebra for cEF_k .*

Proof. (\Leftarrow) Suppose that $\mathfrak{S}(L)$ is an algebra for cEF_k . By Theorem A.10, every language recognised by $\mathfrak{S}(L)$ is invariant under \sim_k^m , for some m (when considering regular forests only). Consequently, the claim follows by Corollary A.4.

(\Rightarrow) If L is cEF_k -definable, it follows by Corollary A.4 that L is \sim_k^m -invariant, for some m . Thus \sim_k^m is contained in the syntactic congruence of L , which means that the syntactic morphism $\eta : \mathbb{F}\Sigma \rightarrow \mathfrak{S}(L)$ maps \sim_k^m -equivalent forests to the same value. Given forests $s, t \in \mathbb{F}(S_0 \cup S_1)$ with $s \sim_k^m t$, we can choose forests $s', t' \in \mathbb{F}\Sigma$ with $s' \sim_k^m t'$ and $s(v) = \eta(s'(v))$ and $t(v) = \eta(t'(v))$. Thus,

$$s \sim_k^m t \quad \text{implies} \quad \pi(s) = \eta(s') = \eta(t') = \pi(t).$$

By Theorem A.10, it follows that $\mathfrak{S}(L)$ is an algebra for cEF_k . ◀

Constructing Deterministic ω -Automata from Examples by an Extension of the RPNI Algorithm

León Bohn  

RWTH Aachen University, Germany

Christof Löding 

RWTH Aachen University, Germany

Abstract

The RPNI algorithm (Oncina, Garcia 1992) constructs deterministic finite automata from finite sets of negative and positive example words. We propose and analyze an extension of this algorithm to deterministic ω -automata with different types of acceptance conditions. In order to obtain this generalization of RPNI, we develop algorithms for the standard acceptance conditions of ω -automata that check for a given set of example words and a deterministic transition system, whether these example words can be accepted in the transition system with a corresponding acceptance condition. Based on these algorithms, we can define the extension of RPNI to infinite words. We prove that it can learn all deterministic ω -automata with an informative right congruence in the limit with polynomial time and data. We also show that the algorithm, while it can learn some automata that do not have an informative right congruence, cannot learn deterministic ω -automata for all regular ω -languages in the limit. Finally, we also prove that active learning with membership and equivalence queries is not easier for automata with an informative right congruence than for general deterministic ω -automata.

2012 ACM Subject Classification Theory of computation \rightarrow Automata over infinite objects

Keywords and phrases deterministic omega-automata, learning from examples, learning in the limit, constructing acceptance conditions, active learning

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.20

Related Version *Full Version:* <https://arxiv.org/abs/2108.03735>

1 Introduction

In this paper we consider learning problems for automata on infinite words, also referred to as ω -automata, which have been studied since the early 1960s as a tool for solving decision problems in logic [7] (see also [26]), and are nowadays used in procedures for formal verification and synthesis of reactive systems (see, e.g., [5, 27, 18] for surveys and recent work). Syntactically ω -automata are very similar to NFA resp. DFA (standard nondeterministic resp. deterministic finite automata on finite words), and they also share many closure and algorithmic properties. However, many algorithms and constructions are much more involved for ω -automata, one prominent such example being determinization [23, 22, 24, 15], and another one the minimization of deterministic ω -automata [25], which is hard for most of the acceptance conditions of ω -automata. The underlying reason is that regular languages of finite words have a simple characterization in terms of the Myhill/Nerode congruence, and the unique minimal DFA for a regular language can be constructed by merging language equivalent states (see [13]). In contrast, deterministic ω -automata need, in general, different language equivalent states for accepting a given regular ω -language.

The characterization of minimal DFA in terms of the Myhill/Nerode congruence is also an important property that is used by learning algorithms for DFA. In automaton learning one usually distinguishes the two settings of passive and active learning. We are mainly concerned with passive learning in this paper, where the task is to construct an automaton



© León Bohn and Christof Löding;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 20; pp. 20:1–20:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

from a sample, a given finite set of words together with a classification if they are in the language or not. The RPNI algorithm [21] is a passive learning algorithm that constructs a DFA from a given sample of positive and negative examples (words that are in the language and words that are not in the language, respectively). It starts with the prefix tree acceptor, a tree shaped DFA that accepts precisely the positive examples and subsequently it tries to merge pairs of states in the canonical order of words (each state is associated with the word reaching it in the prefix tree acceptor). If a merge results in a DFA that accepts a negative example, the merge is discarded. Otherwise the merge is kept and the algorithm continues with this DFA. RPNI can learn the minimal DFA for each regular language in the limit with polynomial time and data. This means that RPNI runs in polynomial time in the size of the given sample, and for each regular language L there is a characteristic sample S_L of polynomial size, such that RPNI produces the minimal DFA for L for each sample that is consistent with L and contains S_L [21]. The RPNI algorithm is a simple algorithm that also produces useful results if the sample does not include the characteristic sample of any language L . Therefore its principle of state merging has been used for other automaton models, e.g., probabilistic automata [8, 17] and sequential transducers [20].

In this paper we propose and analyze an extension of RPNI to ω -automata. In the setting of infinite words, one uses ultimately periodic words of the form uv^ω for finite words u, v . These are infinite words with a finite representation, and each regular ω -language is uniquely determined by the set of ultimately periodic words that it contains (see [26]). There are two main obstacles that one has to overcome for a generalization of RPNI. First, it is not clear how to generalize the prefix tree acceptor to infinite words, since a tree shaped acceptor for a set of infinite words necessarily needs to be infinite. We therefore propose a formulation of the algorithm that inserts transitions instead of merging states, and creates new states in case none of the existing states can be used as a target of the transition. In the setting of finite words, this method of inserting transitions produces the same result as RPNI, and it can easily be used for infinite words as well. Because this algorithm produces growing transition system, we call it **Sprout**.

The second problem arises in the test whether a merge (in our formulation an inserted transition) should be kept or discarded. In the case of finite words, one can simply check whether there are a positive and a negative example that reach the same state, which obviously is not possible in a DFA that is consistent with the sample. For ω -automata the situation is a bit more involved, because acceptance of a word is not determined by a single state, but rather the set of states that is reached infinitely often. And furthermore, there are various acceptance conditions using different ways of classifying these infinity sets into accepting and rejecting. To solve this problem, we propose polynomial time algorithms for checking whether a deterministic transition system admits an acceptance condition of a given type (Büchi, generalized Büchi, parity, or Rabin) that turns the transition system into a deterministic ω -automaton that is consistent with the sample. These consistency algorithms are then used in **Sprout** in order to check whether a merge (inserted transition) produces a transition system that can still be consistent with the sample (for the acceptance condition under consideration). However, we believe that these consistency algorithms are of interest on their own and might also be useful in other contexts. We also show that bounding the size of the acceptance condition can make the problem hard: consistency with a Rabin condition with three pairs or generalized Büchi condition with three sets is NP-hard.

Our analysis of **Sprout** reveals that it can learn every ω -regular language with an informative right congruence (IRC) in the limit from polynomial time and data. A deterministic ω -automaton has an informative right congruence if it has only one state for each My-

hill/Nerode equivalence class of the language that it defines [3]. Recently, another algorithm that can learn every ω -regular language with an IRC in the limit from polynomial time and data has been proposed [4]. This algorithm is an extension of the approach from [12] from finite to infinite words. However, the algorithm from [4] has explicitly been developed for automata with an IRC, and it can only produce such automata (it defaults to an automaton accepting precisely the positive examples in case the sample does not completely characterize the target automaton). In contrast, **Sprout** is not specifically designed for IRC languages, it can also construct automata that do not have an IRC. But on the negative side we also show that **Sprout** cannot learn a deterministic ω -automaton for every regular ω -language.

The positive results for passive learning of IRC languages raise the question whether this class is also simpler for active learning than general deterministic ω -automata. The standard model for active learning of automata uses membership and equivalence queries, and DFA can be learned in polynomial time in this model [1]. This approach has been extended to the class of weak deterministic Büchi automata [16], whose minimal automata can also be defined using the standard right congruence. For general regular ω -languages, the only known algorithms either learn a different representation based on DFA [2], or add another query about the loop structure of the target automaton [19]. Since the characterization of the minimal automata by a right congruence is a crucial point in many active learning algorithms, it is tempting to believe that the algorithms can be extended to the classes of languages with an IRC. We prove that this is not the case by showing that a polynomial time active learning algorithm for deterministic ω -automata with an IRC can be turned into a polynomial time learning algorithm for general deterministic ω -automata.

Finally, we also make the observation that polynomial time active learning (with membership and equivalence queries) is at least as hard as learning in the limit with polynomial time and data.

The paper is structured as follows. In Section 2 we give basic definitions. In Section 3 we present the consistency algorithms, and in Section 4 we describe our extension of RPNI to ω -automata. In Section 5 we show that the property of an IRC does not help for polynomial time active learning, and in Section 6 we conclude.

2 Preliminaries

For a finite alphabet Σ we use Σ^* and Σ^ω to refer to the set of finite and infinite words respectively. The empty word is denoted by ε , and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. A deterministic transition system (TS) is defined by a tuple $\mathcal{T} = (Q, \Sigma, q_0, \delta)$ where Q is a finite set of states, Σ a finite alphabet, $q_0 \in Q$ the initial state and $\delta : Q \times \Sigma \rightarrow Q$ is the transition function. We use $\delta(q, a) = \perp$ to indicate that a transition $(q, a) \in Q \times \Sigma$ is not defined in \mathcal{T} . Further we extend δ to $\delta^* : Q \times \Sigma^* \rightarrow Q$ defined as $\delta^*(q, \varepsilon) = q$ and $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$ for $q \in Q, a \in \Sigma$ and $w \in \Sigma^*$. Unless otherwise specified \mathcal{T} will be used to refer to a transition system with components as above. The unique run of \mathcal{T} on $w \in \Sigma^\omega$ is a sequence of transitions $\rho = q_0 w_0 q_1 w_1 \dots$ with $q_{i+1} = \delta(q_i, w_i)$. For an infinite run ρ we denote by $\text{inf}(\rho)$ the *infinity set* of ρ , consisting of all state-symbol pairs that occur infinitely often in ρ . A set of states $\emptyset \neq C \subseteq Q$ is called *strongly connected* if for all $p, q \in C$ we have $\delta^*(p, w) = q$ for some $w \in \Sigma^+$. The \subseteq -maximal strongly connected sets of \mathcal{T} are called strongly connected components (SCCs) and for a set R we use $\text{SCC}(R)$ to refer to the set of all SCCs $S \subseteq R$.

Augmenting a transition system \mathcal{T} with an acceptance condition \mathcal{C} yields an ω -automaton $\langle \mathcal{T}, \mathcal{C} \rangle = (Q, \Sigma, q_0, \delta, \mathcal{C})$. We now introduce different types of acceptance conditions (based on the survey [26]), give a notion of their size $|\mathcal{C}|$ and define which sets $X \subseteq Q \times \Sigma$ satisfy

them. Note that while acceptance is often defined based on *states* that occur infinitely often, we opt for transition-based acceptance due to its succinctness (state-based acceptance can be turned into transition-based acceptance without changing the transition system, while the transformation in the other direction requires a blow-up of the transition system depending on the acceptance condition).

A *Büchi condition* $F \subseteq Q \times \Sigma$ is satisfied if $X \cap F \neq \emptyset$, whereas a *generalized Büchi condition* $\mathcal{B} = \{F_1, \dots, F_k\}$ with $F_i \subseteq Q \times \Sigma$ is satisfied if $X \cap F_i \neq \emptyset$ for all $i \in [1, k] \subseteq \mathbb{N}$. The set X satisfies a *parity condition* $\kappa : (Q \times \Sigma) \rightarrow C$ for a finite $C \subseteq \mathbb{N}$ if $\min(\kappa(X))$ is even where $\kappa(X) = \{\kappa(q, a) : (q, a) \in X\}$. We call $\mathcal{R} = \{(E_1, F_1), \dots, (E_k, F_k)\}$ with $E_i, F_i \subseteq (Q \times \Sigma)$ a *Rabin condition* and it is satisfied if $E_i \cap X = \emptyset$ and $F_i \cap X \neq \emptyset$ for some $i \in [1, k] \subseteq \mathbb{N}$. Finally a *Muller condition* $\mathcal{F} \subseteq 2^{Q \times \Sigma}$ is satisfied if $X \in \mathcal{F}$. For an acceptance condition \mathcal{C} of type $\Omega \in \{\text{Parity, generalized Büchi, Rabin}\}$ we use $|\mathcal{C}|$ to refer to the number of priorities/recurring sets/Rabin pairs respectively. We use abbreviations (g)DBA, DPA, DRA to refer to deterministic (generalized) Büchi, Parity and Rabin automata and introduce a set Acc containing these acceptance types. An automaton $\mathcal{A} = \langle \mathcal{T}, \mathcal{C} \rangle$ accepts $w \in \Sigma^\omega$ if $\text{inf}(\rho)$ satisfies \mathcal{C} , where ρ refers to the unique run of \mathcal{T} on w . The set of all words that are accepted by \mathcal{A} is the language accepted by \mathcal{A} , denoted by $L(\mathcal{A})$.

Let \sim be an equivalence relation over Σ^* . We refer to the equivalence class of x under \sim as $[x]_\sim = \{y \in \Sigma^* : x \sim y\}$ and call \sim a (right) *congruence* if $u \sim v$ implies $ua \sim va$ for all $a \in \Sigma$. A regular language $L \subseteq \Sigma^\omega$ induces the *canonical right congruence* \sim_L in which $u \sim_L v$ holds if and only if $u^{-1}L = v^{-1}L$ with $u^{-1}L = \{w \in \Sigma^\omega : uw \in L\}$. Using the terminology of [3], we say that an automaton \mathcal{A} has an *informative right congruence* (IRC) if $u \sim_{L(\mathcal{A})} v$ implies that \mathcal{A} reaches the same state from q_0 when reading u or v . A language L has an Ω -IRC for $\Omega \in \text{Acc}$ if an Ω -automaton with an IRC which recognizes L exists and we denote by $\text{ind}(L)$ the number of equivalence classes of \sim_L .

A word $w \in \Sigma^\omega$ is called *ultimately periodic* if $w = uv^\omega$ with $u \in \Sigma^*, v \in \Sigma^+$. We denote by UP_Σ the set of all ultimately periodic words in Σ^ω and note that two regular languages $K, L \subseteq \Sigma^\omega$ are equal if and only if $K \cap \text{UP}_\Sigma = L \cap \text{UP}_\Sigma$ [7]. Note that there always exists a reduced form $w = uv^\omega$ in which u and v are as short as possible. We call a pair $S = (S_+, S_-)$ with $S_+, S_- \subseteq \text{UP}_\Sigma$ and $S_+ \cap S_- = \emptyset$ a *sample* and say that S is *in reduced form* if each $uv^\omega \in S$ is in a reduced form where $w \in S$ is used as a shorthand for $w \in S_+ \cup S_-$. For $L \subseteq \Sigma^\omega$ we say that S is consistent with L if $S_+ \subseteq L$ and $S_- \cap L = \emptyset$. Similarly an automaton \mathcal{A} is consistent with S if $S_+ \subseteq L(\mathcal{A})$ and $S_- \cap L(\mathcal{A}) = \emptyset$.

For $\Omega \in \text{Acc}$ we call a function f that maps a sample to an Ω -automaton a *passive learner*. f is called *consistent* if for any sample S the constructed automaton $f(S)$ is consistent with S . A sample S_L is *characteristic* for L and f if for any sample S that is consistent with L and that contains S_L , the learner produces an automaton $f(S)$ recognizing L . For a class of representations of languages \mathbb{C} (in our case deterministic ω -automata) we use $\mathcal{L}(\mathbb{C})$ to refer to the represented languages and define the size of $L \in \mathcal{L}(\mathbb{C})$ to be the size of the minimal representation of L in \mathbb{C} . Based on the definition in [10] we say \mathbb{C} is *learnable in the limit using polynomial time and data* if there exists a learner f that runs in polynomial time for any input sample, and for each $L \in \mathcal{L}(\mathbb{C})$ there exists a characteristic sample whose size is polynomial in the size of L .

We call $w \in \Sigma^\omega$ *escaping* from $p \in Q$ with $a \in \Sigma$ in \mathcal{T} if there exists a decomposition $w = uav$ with $v \in \Sigma^\omega$ such that $\delta^*(q_0, u) = p$ and $\delta(p, a) = \perp$. We refer to ua as the *escape-prefix* and call av the *exit string* of w . Two escaping words w_1, w_2 are *indistinguishable* if they escape \mathcal{T} from the same state and their exit strings coincide. We call \mathcal{T} Ω -*consistent* with a sample S if there exists an Ω -acceptance condition \mathcal{C} such that $\{w \in S_+ : w \text{ not escaping in } \mathcal{T}\} \subseteq$

$L(\mathcal{T}, \mathcal{C}), S_- \cap L(\mathcal{T}, \mathcal{C}) = \emptyset$ and no pair of sample words from $S_+ \times S_-$ is indistinguishable. Note that Ω -consistency with a transition system does not require all words from S_+ to have an infinite run in the transition system. It just means that \mathcal{T} does not produce any conflicts between words in S_+ and in S_- . In contrast, for an automaton to be considered consistent with S it is required that all words from S_+ are accepted.

3 Consistency Algorithms

The algorithm for learning ω -automata that we describe in Section 4 constructs a transition system and then tests whether an acceptance condition can be found such that all sample words are accepted and rejected accordingly. In this section we develop algorithms for this test, so we assume that a transition system $\mathcal{T} = (Q, \Sigma, q_0, \delta)$ is given. We do not work with the sample directly in this section, and rather work with the infinity sets induced by the sample words. This leads to the notion of a partial condition, which we define below. Then we investigate how different types of acceptance conditions that are consistent with such a partial condition can be constructed.

Recall that a Muller condition $\mathcal{F} \subseteq 2^{Q \times \Sigma}$ is satisfied by an infinity set $X \subseteq Q \times \Sigma$ if and only if $X \in \mathcal{F}$. Instead of specifying such a Muller condition based solely on the infinity sets that satisfy it, we can also define it as a partition $\mathcal{F} = (\mathcal{F}_0, \mathcal{F}_1)$ of $2^{Q \times \Sigma}$ into accepting and rejecting sets, in the following also referred to as positive and negative sets respectively. In other words such a condition assigns to each possible set $X \subseteq Q \times \Sigma$ a *classification* $\sigma \in \{0, 1\}$, which we denote as $\mathcal{F}(X) = \sigma$ for $X \in \mathcal{F}_\sigma$. Note that any acceptance condition \mathcal{C} can be viewed as a Muller condition $(\mathcal{F}_0^{\mathcal{C}}, \mathcal{F}_1^{\mathcal{C}})$ by assigning to $\mathcal{F}_0^{\mathcal{C}}$ exactly those sets $X \subseteq Q \times \Sigma$ that satisfy \mathcal{C} and defining $\mathcal{F}_1^{\mathcal{C}}$ to contain all others.

To incorporate the fact that the infinity sets induced by sample words might not classify all subsets of $Q \times \Sigma$, we introduce the concept of a *partial condition* $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$ with $\mathcal{H}_0, \mathcal{H}_1 \subseteq 2^{Q \times \Sigma}$ in which only a subset of all elements $X \subseteq Q \times \Sigma$ receives a classification $\mathcal{H}(X) \in \{0, 1\}$. We use $X \in \mathcal{H}$ to denote $X \in \mathcal{H}_0 \cup \mathcal{H}_1$ and call a partial condition $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$ *consistent* if $\mathcal{H}_0 \cap \mathcal{H}_1 = \emptyset$. A component \mathcal{H}_σ of a partial condition is called *union-closed* if for any finite collection $X_1, \dots, X_n \in \mathcal{H}_\sigma$ we have $X_1 \cup \dots \cup X_n \notin \mathcal{H}_{1-\sigma}$ or in other words the union of positive sets is not negative and vice versa. We call an acceptance condition \mathcal{C} *consistent with* a partial condition $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$ if $\mathcal{H}_0 \subseteq \mathcal{F}_0^{\mathcal{C}}$ and $\mathcal{H}_1 \subseteq \mathcal{F}_1^{\mathcal{C}}$.

For each $\Omega \in \text{Acc}$ we can now define the decision problem Ω -CONSISTENCY: Given a transition system $\mathcal{T} = (Q, \Sigma, q_0, \delta)$ and a partial condition $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$ with $\mathcal{H}_0, \mathcal{H}_1 \subseteq 2^{Q \times \Sigma}$, the question is whether there exists an acceptance condition \mathcal{C} of type Ω over $Q \times \Sigma$ that is consistent with \mathcal{H} . In the following we provide algorithms that decide Ω -CONSISTENCY for the various acceptance types we introduced and investigate their complexity.

Büchi and generalized Büchi conditions. For a Büchi condition $F \subseteq Q \times \Sigma$ we know that every superset of some $X \subseteq Q \times \Sigma$ with $X \cap F \neq \emptyset$ clearly has a non-empty intersection with F . Based on this observation we can define an algorithm that computes for a given partial condition $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$ a Büchi condition F which is consistent with \mathcal{H} . We forego a formal definition of the algorithm itself and instead define the partial function it computes, where a result of \perp is used to indicate that no Büchi condition exists that is consistent with \mathcal{H} .

$$\text{BuchiCons}(\mathcal{H}_0, \mathcal{H}_1) = \begin{cases} \text{return } \perp & \text{if } P \in \mathcal{H}_0 \text{ exists with } P \subseteq \bigcup \mathcal{H}_1 \\ \text{return } (Q \times \Sigma) \setminus \bigcup \mathcal{H}_1 & \text{otherwise} \end{cases}$$

It is easily verified that BuchiCons is computable in polynomial time and a formal proof for the correctness of this algorithm can be found in the full version of this paper.

With generalized Büchi conditions it is no longer guaranteed that the union of two negative sets $N, N' \in \mathcal{H}_1$ is also negative. Consider a generalized Büchi condition $\mathcal{B} = \{F, F'\}$ such that N has a non-empty intersection with F but not with F' , whereas $N' \cap F = \emptyset$ and $N' \cap F' \neq \emptyset$. Then their union $F \cup F'$ has a non-empty intersection with both F and F' and hence satisfies \mathcal{B} . Therefore we first isolate the \subseteq -maximal sets N_1, \dots, N_k in \mathcal{H}_1 . As before we give a function

$$\text{genBuchiCons}(\mathcal{H}_0, \mathcal{H}_1) = \begin{cases} \text{return } \perp & \text{if } P \in \mathcal{H}_0 \text{ with } P \subseteq N_i \text{ exists} \\ \text{return } \{(Q \times \Sigma) \setminus N_i : i \leq k\} & \text{otherwise} \end{cases}$$

which maps a partial condition to a generalized Büchi condition that is consistent with \mathcal{H} or \perp if no such condition exists. It is again not difficult to see that an algorithm can compute genBuchiCons in polynomial time. A formal proof of the correctness of genBuchiCons can be found in the full version of this paper.

► **Theorem 1.** *The algorithm (*gen*)BuchiCons decides the (generalized) Büchi-CONSISTENCY problem in polynomial time and returns a corresponding acceptance condition if one exists.*

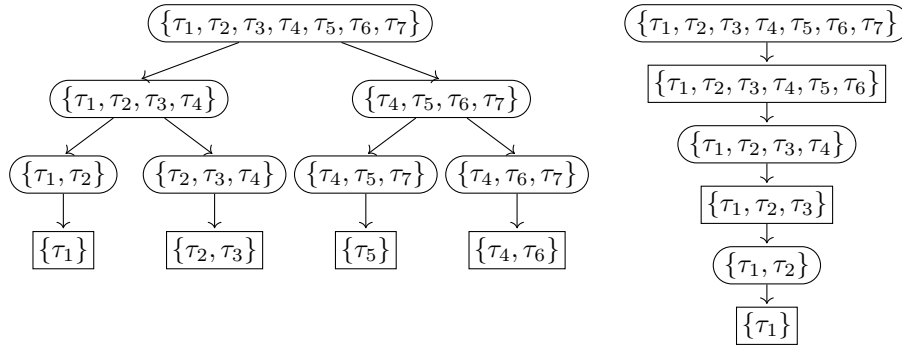
Parity Conditions. It is a well-known observation that for a given Muller condition $(\mathcal{F}_0, \mathcal{F}_1)$ there exists an equivalent parity condition κ if and only if \mathcal{F}_0 and \mathcal{F}_1 are union-closed [28]. We show an analogous statement for partial conditions, starting with the following lemma which establishes that if the union of positive and negative elements coincide, then no equivalent parity condition can be found.

► **Lemma 2.** *Let $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$ be a consistent partial condition. If we have $P = N$ for $P = P_1 \cup \dots \cup P_k$ and $N = N_1 \cup \dots \cup N_l$ with $P_i \in \mathcal{H}_0$ and $N_j \in \mathcal{H}_1$ then there exists no parity condition that is consistent with \mathcal{H}*

It turns out that the opposite direction also holds, meaning if no such unions of positive and negative sets can be found, then an equivalent parity condition must exist. This implication arises as a consequence of the ParityCons algorithm we present later together with the proofs of its correctness. For a given partial condition ParityCons (see Algorithm 1) attempts to construct a chain of sets of transitions $Z_0 \supseteq Z_1 \supseteq \dots \supseteq Z_{n-1}$ with alternating classifications $\sigma_i \in \{0, 1\}$, i.e., $\sigma_{i+1} = 1 - \sigma_i$ for $i < n - 1$. We refer to this as a *Zielonka path* because it corresponds to the Split or Zielonka tree representation of a parity condition [28, 11].

From such a Zielonka path one obtains a parity condition κ where $\kappa(q, a) = \sigma_0 + i$ for the maximal i such that $(q, a) \in Z_i$. On the other hand every parity condition κ with priorities C determines a chain $Z_0 \supseteq Z_1 \supseteq \dots \supseteq Z_{|C|-1}$ and alternating classifications σ_i where $\sigma_0 = \min(C) \bmod 2$, $\sigma_{i+1} = 1 - \sigma_i$ and Z_i contains all state-symbol pairs whose color is greater or equal to $\sigma_0 + i$. To guarantee the existence of such an alternating chain, we assume that κ is optimal and contains no gaps, which can be ensured in polynomial time [9].

► **Example 3.** As an example consider a partial condition $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$ with set inclusion diagram as shown on the left of Figure 1, where \mathcal{H}_0 contains the transition sets drawn with rounded border, and \mathcal{H}_1 those with rectangular border (the leaves of the tree, in this example). We assume an underlying transition system in which the transition sets in \mathcal{H} are strongly connected. It is easily verified that \mathcal{H} does not satisfy the condition of Lemma 2. Since we claimed the converse of Lemma 2 to be true, a parity condition that is consistent with \mathcal{H} should exist. It turns out that such a parity condition requires 6 distinct priorities (the corresponding Zielonka path is shown on the right of Figure 1) even though there is at most one alternation between positive and negative sets along inclusion chains in \mathcal{H} . This is due to the fact that more alternations are introduced by unions of positive and negative sets.



■ **Figure 1** On the left an inclusion graph for the partial condition \mathcal{H} from Example 3 can be seen in which positive elements are depicted with rounded and negative ones with rectangular borders. The path depicted on the right corresponds to a priority function κ with domain $\{0, 1, 2, 3, 4, 5\}$ such that $\tau_7 \mapsto 0, \tau_6 \mapsto 1, \tau_5 \mapsto 1, \tau_4 \mapsto 2, \tau_3 \mapsto 3, \tau_2 \mapsto 4, \tau_1 \mapsto 5$ which is the minimal parity condition that is consistent with \mathcal{H} .

We now present an algorithm that given a consistent partial condition \mathcal{H} over $Q \times \Sigma$ constructs an equivalent parity condition with the least number of distinct priorities if one exists. As a simplification we assume that the set $Q \times \Sigma$ of all transitions is classified by \mathcal{H} , which enables us to use $Q \times \Sigma$ as the first set Z_0 of the chain that is constructed. We describe later how partial conditions that do not satisfy this assumption can be dealt with.

■ **Algorithm 1** ParityCons.

Input: A consistent partial condition $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$ with $Q \times \Sigma \in \mathcal{H}$

Output: A Zielonka path $(Z_0, \sigma_0), (Z_1, \sigma_1), \dots, (Z_{n-1}, \sigma_{n-1})$

$Z_0 \leftarrow Q \times \Sigma, \sigma_0 \leftarrow \mathcal{H}(Q \times \Sigma), i \leftarrow 0$

repeat

$i \leftarrow i + 1$

$Z \leftarrow \bigcup \{X \subseteq Z_{i-1} : \mathcal{H}(X) = 1 - \sigma_{i-1}\}$

if $Z = Z_{i-1}$ **then**

return *No consistent parity condition exists.*

$Z_i \leftarrow Z, \sigma_i \leftarrow 1 - \sigma_{i-1}$

until $Z = \emptyset$

return $(Z_0, \sigma_0), \dots, (Z_{i-1}, \sigma_{i-1})$

After Z_0 and its corresponding classification $\sigma_0 = \mathcal{H}(Z_0)$ have been determined, the algorithm computes Z_1 as the union of all $1 - \sigma_0$ subsets of Z_0 . If this union coincides with Z_0 then the conditions for Lemma 2 are met and the algorithm terminates prematurely as no equivalent parity condition can exist. Otherwise this construction ensures that every strict superset of Z_1 receives the same classification as Z_0 from the constructed parity condition. This process is then repeated for Z_1 with $\sigma_1 = 1 - \sigma_0$, Z_2 with $\sigma_2 = 1 - \sigma_1$ and so on until no subsets of opposite classification remain. At this point the algorithm terminates and returns the constructed chain of sets of transitions together with their corresponding classification.

Proving the correctness of this approach forms the opposite direction of Lemma 2 as it entails that if no union of positive and negative sets as in Lemma 2 is found, an equivalent parity condition can be constructed. One restriction on the partial conditions that can be passed to ParityCons is that the set of all transitions, $Q \times \Sigma$, must be present in either \mathcal{H}_0 or \mathcal{H}_1 . As these partial conditions arise from the infinity sets that words from a finite sample

induce, however, it is easily conceivable that there are many scenarios - for example when the automaton that we want to learn is made up of multiple SCCs - in which no word inducing $Q \times \Sigma$ exists. In this case we can simply define two extended partial conditions \mathcal{H}^p and \mathcal{H}^n in which $Q \times \Sigma$ is added as a positive or negative set respectively and execute `ParityCons` separately for each of them. If only one computation results in a Zielonka path we are done, otherwise the two resulting paths are compared with regard to their length and the longer one is discarded.

► **Theorem 4.** *ParityCons decides Parity-CONSISTENCY in polynomial time and returns a corresponding parity condition with a minimal number of priorities if one exists.*

Proof (sketch). We proceed in two steps and first show that the classification obtained by the Zielonka path computed by `ParityCons` are indeed consistent with the original partial condition. Subsequently one shows that if the computation exits prematurely, then there exist positive and negative sets whose unions coincide, which by Lemma 2 means that no equivalent parity condition exists. ◀

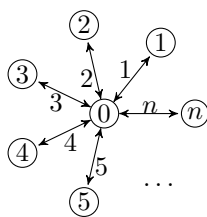
Rabin Conditions. We now turn towards computing an equivalent Rabin condition based on a given partial condition, for which we again utilize an observation about union-closedness. Specifically, a Muller condition is equivalent to a Rabin condition if and only if \mathcal{F}_1 is union-closed [28]. The algorithm `RabinCons` (see Algorithm 2) computes for each positive set P in \mathcal{H} a separate Rabin pair (E_P, F_P) in which each transition that is not part of P belongs to E_P and every transition which does not occur in a negative subloop of P belongs to F_P . In case a positive loop is equal to the union of its maximal negative subloops, no equivalent Rabin condition can be found as the condition on union-closedness outlined above is violated.

■ **Algorithm 2** `RabinCons`.

Input: A consistent partial condition $\mathcal{H} = (\mathcal{H}_0, \mathcal{H}_1)$
Output: A Rabin condition \mathcal{R} consistent with \mathcal{H}
 $\mathcal{R} \leftarrow \emptyset$
foreach $P \in \mathcal{H}_0$ **do**
 $N_1, \dots, N_k \leftarrow$ maximal sets in $\mathcal{P}(P) \cap \mathcal{H}_1$
 $E_P \leftarrow (Q \times \Sigma) \setminus P$
 $F_P \leftarrow P \setminus (N_1 \cup \dots \cup N_k)$
 if $F_P = \emptyset$ **then**
 | **return** *No consistent Rabin condition exists*
 $\mathcal{R} \leftarrow \mathcal{R} \cup \{(E_P, F_P)\}$
return \mathcal{R}

► **Theorem 5.** *The algorithm RabinCons decides Rabin-CONSISTENCY in polynomial time and returns a corresponding Rabin condition if one exists.*

A Rabin condition produced by `RabinCons` has $|\mathcal{H}_0|$ pairs and is not guaranteed to have the minimal number of pairs. Even though it is possible to find optimizations which might make use of the underlying structure with regard to strongly connected components and subset relations between positive and negative loops, we now illustrate why the computation of an optimal Rabin condition (with a minimal number of pairs) is NP-hard.



■ **Figure 2** This figure contains a depiction of the transition system $\mathcal{T}_{\mathcal{G}}$, which can be used to show NP-completeness of k -generalized Büchi-CONSISTENCY and k -Rabin-CONSISTENCY.

Fixed-size consistency. For each acceptance type $\Omega \in \{\text{generalized Büchi, Parity, Rabin}\}$ and every natural number $k \in \mathbb{N}$ we define the decision problem k - Ω -CONSISTENCY: Given a transition system \mathcal{T} and a consistent partial condition \mathcal{H} the question is whether there is an acceptance condition \mathcal{C} of type Ω in \mathcal{T} which is consistent with \mathcal{H} such that $|\mathcal{C}| \leq k$. The algorithm `ParityCons` we provided earlier decides k -Parity-CONSISTENCY in polynomial time, however finding a Rabin or generalized Büchi condition of bounded size turns out to be much more difficult.

Intuitively, the difficulty in finding an optimal generalized Büchi condition with at most k components arises from the fact that the union of two negative sets is not necessarily guaranteed to also be negative. As there are in general exponentially many possible ways of partitioning the transitions into k sets, a procedure for constructing an optimal generalized Büchi condition would need to consider all of them. In the following we establish that the fixed-size consistency problem for generalized Büchi conditions is already NP-complete when $k = 3$. This is done by giving a reduction from 3-Coloring for directed graphs, which is known to be NP-complete [14].

► **Lemma 6.** *3-generalized Büchi-CONSISTENCY is NP-complete.*

Proof. Let $\mathcal{G} = (V, E)$ be a finite directed graph with $V = \{v_1, v_2, \dots, v_n\}$. We define the deterministic partial transition system

$$\mathcal{T}_{\mathcal{G}} = (\{0, 1, 2, \dots, n\}, \{1, 2, \dots, n\}, 0, \delta) \text{ with } \delta(q, a) = \begin{cases} a & \text{if } q = 0 \\ 0 & \text{if } q = a \\ \perp & \text{otherwise} \end{cases}$$

which is depicted in Figure 2. Note that it is possible to construct an equivalent transition system over a binary alphabet $\Sigma' = \{a, b\}$ by encoding $i \in \Sigma$ as $a^i b$. Thus our choice of Σ depending on the size of the graph merely serves to simplify notation in the following. We define a sample $S_{\mathcal{G}} = (P_{\mathcal{G}}, N_{\mathcal{G}})$ with

$$P_{\mathcal{G}} = \{p_{ij} : (v_i, v_j) \in E\} \text{ and } N_{\mathcal{G}} = \{n_i : 0 < i \leq n\} \text{ where } p_{ij} = (iij)^\omega, n_i = i^\omega$$

In the following we use \bar{p}_{ij} and \bar{n}_i to refer to the infinity set of the unique run of $\mathcal{T}_{\mathcal{G}}$ on p_{ij} and n_i respectively. Let $c : V \rightarrow \{1, 2, 3\}$ be a 3-coloring for V such that $c(v_i) \neq c(v_j)$ for all $(v_i, v_j) \in E$. We construct a generalized Büchi condition $\mathcal{B}_{\mathcal{G}} = (F_1, F_2, F_3)$ with $F_k = \{i : c(v_i) \neq k\}$, witnessing membership in 3-generalized Büchi-CONSISTENCY. For all $i \leq n$ we have for $k = c(v_i)$ that $\bar{n}_i \cap F_k = \{0, i\} \cap F_k = \emptyset$ and thus $n_i \notin L(\mathcal{T}, \mathcal{B}_{\mathcal{G}})$. On the other hand $\bar{p}_{ij} \cap F_k = \{0, i, j\} \cap F_k \neq \emptyset$ for all k as $c(v_i) \neq c(v_j)$ is guaranteed for all $(v_i, v_j) \in E$ by the coloring function c . Hence $p_{ij} \in L(\mathcal{T}, \mathcal{B}_{\mathcal{G}})$ and the constructed condition is indeed consistent with the sample.

20:10 Constructing Deterministic ω -Automata from Examples

For the other direction assume that there exists a generalized Büchi condition $\mathcal{B} = (F_1, F_2, F_3)$ such that $\langle \mathcal{T}, \mathcal{B} \rangle$ is consistent with S . Clearly it must hold that $F_1 \cap F_2 \cap F_3 = \emptyset$ as otherwise there would exist some word $n_i \in N_{\mathcal{G}}$ with $\bar{n}_i \cap F_k = \{0, i\} \cap F_k \neq \emptyset$ for all k , which would contradict consistency with S . We can now define a coloring $c : V \rightarrow \{1, 2, 3\}$ with $c(v_i) = \min\{k : i \notin F_k\}$. For any $v_i, v_j \in V$ with $c(v_i) = c(v_j) = k$ we have $(v_i, v_j) \notin E$. If not then there would exist a word $p_{ij} \in P_{\mathcal{G}}$ for which consistency guarantees that $\bar{p}_{ij} \cap F_k = \{0, i, j\} \cap F_k \neq \emptyset$, which can only hold if v_i and v_j are assigned different colors. Thus c is indeed a valid 3-coloring, which concludes the reduction proof.

Membership in NP holds as it is possible to verify for a guessed generalized Büchi condition \mathcal{B} of size 3 whether $\langle \mathcal{T}, \mathcal{B} \rangle$ is consistent with S in polynomial time by iterating over all $w \in P_{\mathcal{G}} \cup N_{\mathcal{G}}$ and verifying adequate acceptance/rejection by $\langle \mathcal{T}, \mathcal{B} \rangle$. ◀

A similar reduction can be used to show the NP-hardness of k -Rabin-CONSISTENCY as well. This leads to the following theorem, which establishes the complexity of all fixed-size consistency decision problems we defined above.

► **Theorem 7.** *k -Parity-CONSISTENCY is solvable in polynomial time. For $k > 2$ both k -generalized Büchi-CONSISTENCY and k -Rabin-CONSISTENCY are NP-complete.*

4 Passive learning

Our procedure for the construction of a deterministic partial transition system is inspired by the well known regular positive negative inference (RPNI) algorithm through which deterministic finite automata can be constructed [21]. RPNI first constructs a prefix tree automaton which accepts precisely the positive sample words from S_+ and subsequently attempts to merge states of this automaton in canonical order. If a merge introduces an inconsistency with the sample (i.e. the resulting automaton accepts a word in S_-) it is reverted. Otherwise the algorithm continues with the resulting automaton until no further merges are possible at which point it terminates.

When attempting to transfer this principle to infinite words, it is difficult to find a suitable counterpart for the prefix tree automaton. If we simply attached disjoint loops to the prefix tree at a certain depth, the resulting transition system could certainly be equipped with an acceptance condition such that it accepts precisely S_+ . However, through the introduction of loops with a fixed length that cannot be resolved during the execution, we already determine parts of the structure of the resulting automaton. Instead, we start with a transition system consisting of a single initial state and attempt to introduce new transitions in a specific order (which is reminiscent of the algorithm presented in [6]).

The resulting algorithm **Sprout** is shown in Algorithm 3. In each iteration we begin by computing $\text{Escapes}(S_+, \mathcal{T})$, the set of all prefixes of words in S_+ which are escaping in \mathcal{T} . From this set we now determine the word with the minimal escape-prefix ua in length-lexicographic order. The existing states are then tested as a target for the missing transition in canonical order and if the resulting transition system is Ω -consistent with the sample, we continue with the next escaping word. Checking for consistency is done by using the results from Section 3 and ensuring that no pair of indistinguishable words in $S_+ \times S_-$ exists, both of which are possible in polynomial time. If no suitable target can be found, a new state is introduced instead. See Figure 3 for an illustration. Note that the order in which states are checked as a potential transition target coincides with the order in which merges are attempted in RPNI.

■ **Algorithm 3** Sprout.

Input: A Sample $S = (S_+, S_-)$ over the alphabet Σ and an acceptance type $\Omega \in \text{Acc}$
Output: The deterministic Ω -automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{C})$ consistent with S
 $Q \leftarrow \{\varepsilon\}$, $\delta \leftarrow \emptyset$, $\mathcal{T} \leftarrow (Q, \Sigma, \varepsilon, \delta)$
while $\text{Escapes}(S_+, \mathcal{T}) \neq \emptyset$ **do**
 $ua \leftarrow$ length-lexicographic minimal escape-prefix of a word in S_+
 if $|u| > \text{Thres}(S, \mathcal{T})$ **then**
 | **return** $\text{Aut}(\text{Extend}(Q, \Sigma, \varepsilon, \delta, S_+, S_-), S, \Omega)$
 forall $q \in Q$ *in canonical order* **do**
 | $\delta' \leftarrow \delta \cup \{\hat{u} \xrightarrow{a} q\}$ for the $\hat{u} \in Q$ with $\delta^*(\varepsilon, u) = \hat{u}$
 | **if** $(Q, \Sigma, \varepsilon, \delta')$ *is Ω -consistent with S* **then**
 | $\delta \leftarrow \delta'$ and **continue** with the next escaping word
 $Q \leftarrow Q \cup \{\hat{u}a\}$, $\delta \leftarrow \delta \cup \{\hat{u} \xrightarrow{a} \hat{u}a\}$ for the $\hat{u} \in Q$ with $\delta^*(\varepsilon, u) = \hat{u}$
return $\text{Aut}(\mathcal{T}, S, \Omega)$

Unfortunately there exist samples for which this approach of introducing transitions does not terminate. When executed on $S = (\{(baa)^\omega\}, \{(ab)^\omega, (ba)^\omega, (babaa)^\omega\})$ for example, the algorithm would not terminate and instead construct an infinite b -chain with a -loops on each state. We therefore introduce a threshold on the maximal length of escape-prefixes that are considered in the algorithm. Once this threshold is exceeded, the algorithm terminates. We have chosen the threshold such that we can show completeness for IRC, which works for $\text{Thres}(S, \mathcal{T}) = l_b + l_e^2 + 1$, where l_e and l_b denote the maximal length of u and v for any sample word $uv^\omega \in S$. Intuitively, this value is sufficient to obtain completeness for IRC as any two sample words must have already differed in at least one position once it is exceeded.

If the threshold is exceeded before a transition system is found that is consistent with the sample and has no escaping words from S_+ , the transition system is extended with disjoint loops that guarantee acceptance of the remaining words in S_+ through the function Extend , which we describe in the following. Assume that the algorithm has constructed a transition system $\mathcal{T} = (Q, \Sigma, \varepsilon, \delta)$ for which it then encounters an escape-prefix exceeding the defined threshold. For each state $q \in Q$ we collect all exit strings that leave \mathcal{T} from q in a set E_q . Note that since the shortest escape-prefix in \mathcal{T} exceeded the threshold, each word in E_q must be of the form u^ω for some $u \in \Sigma^+$ and we can write $E_q = \{u_1^\omega, \dots, u_k^\omega\}$.

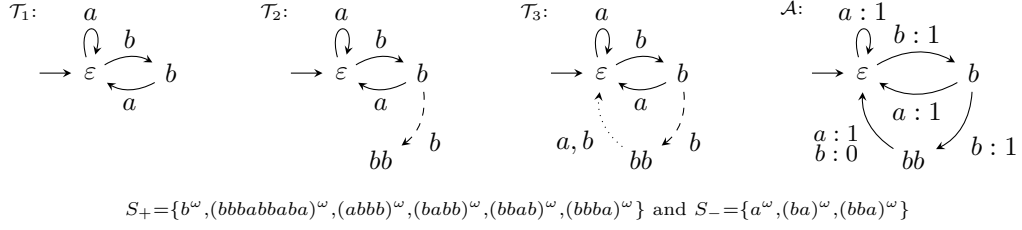
For each state q such that $E_q \neq \emptyset$ we now construct the transition system $\mathcal{T}_{E_q}^\circ$ in which exactly those words that belong to E_q induce loops. To prevent any unintended words from being accepted, we additionally ensure that the initial state of $\mathcal{T}_{E_q}^\circ$ is transient (meaning it cannot be reached from any state within $\mathcal{T}_{E_q}^\circ$). In the following we use $\text{Prf}(u)$ for a word $u \in \Sigma^*$ to denote the set of all prefixes of u . Formally we define $\mathcal{T}_{E_q}^\circ = (Q_{E_q}^\circ, \Sigma, q_0, \delta_{E_q}^\circ)$ with

$$Q_{E_q}^\circ = \{q_0\} \cup \bigcup_{u^\omega \in E_q} \text{Prf}(u)$$

$$\delta_{E_q}^\circ(w, a) = \begin{cases} a & \text{if } w = q_0 \text{ and } a \in \Sigma \cap Q_{E_q}^\circ \\ \varepsilon & \text{if } (wa)^\omega \in E_q \\ wa & \text{if } wa \in Q_{E_q}^\circ \text{ and } (wa)^\omega \notin E_q \\ \perp & \text{otherwise} \end{cases}$$

It is easy to see that q_0 is indeed transient in $\mathcal{T}_{E_q}^\circ$ and we can clearly find a Büchi (and thus also a generalized Büchi, Rabin and Parity) condition such that every word in E_q induces an accepting run in $\mathcal{T}_{E_q}^\circ$. By attaching the corresponding $\mathcal{T}_{E_q}^\circ$ to each state q for which E_q is non-empty, we obtain a transition system in which no word from S_+ is escaping.

20:12 Constructing Deterministic ω -Automata from Examples



■ **Figure 3** In this figure three transition systems that arise during the execution of `Sprout` on the sample $S = (S_+, S_-)$ are depicted. The dashed transition cannot lead to ε as otherwise the union of the infinity sets of $(ba)^\omega$ and $(bba)^\omega$ would coincide with that of $(bbbabbaba)^\omega$ and thus no consistent parity condition exists. Similarly a self-loop on b would mean that the infinity sets induced by $(babb)^\omega$ and $(bba)^\omega$ would coincide. Thus the b -transition must lead to a new state bb . On the right we can see the DPA obtained by augmenting \mathcal{T}_3 with the parity function computed by `ParityCons` on the partial condition induced by S .

Once the main loop terminates, the function `Aut` is called, which uses the results from Section 3 to compute an automaton that is Ω -consistent with S , which is then returned.

► **Proposition 8.** *For a given sample S and an acceptance type $\Omega \in \text{Acc}$ the algorithm `Sprout` computes in polynomial time an automaton of type Ω that is consistent with S .*

While `Sprout` cannot learn all regular ω -languages in the limit (see Proposition 10), we can show completeness for languages with an IRC.

► **Theorem 9.** *The algorithm `Sprout` learns every Ω -IRC language L for $\Omega \in \text{Acc}$ in the limit with polynomial time and data.*

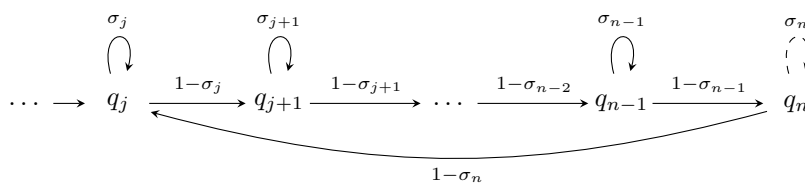
Proof (sketch). We describe the properties that a sample $S = (S_+, S_-)$ has to satisfy in order to be *characteristic* for an Ω -IRC language L :

- The set of prefixes of S_+ has to contain for each \sim_L equivalence class the minimal word in length-lexicographic order on which it is reached.
- Further the sample needs to contain words with which all pairs of equivalence classes can be separated.
- Finally S needs to contain sufficient information about the acceptance condition of an automaton recognizing L .

The first two requirements can be satisfied in a similar way as for the original RPNI algorithm [21]. For parity conditions this has already been investigated in [4]. Below we give a description for Rabin conditions. Detailed definitions for the remaining types of acceptance conditions we introduced can be found in the full version of this paper.

A sample $S_{\mathcal{R}} = (S_+, S_-)$ capturing a Rabin condition \mathcal{R} can be obtained as follows: For each pair (E_i, F_i) we remove all transitions in E_i from the transition system that \mathcal{R} is defined in, decompose the result into its SCCs C_1, \dots, C_k and compute sets K_i consisting of all transitions in C_i . If the set of all transitions K_i in such an SCC satisfies \mathcal{R} we add a word w_i inducing K_i to S_+ , otherwise w_i is added to S_- . For each accepting K_i we then remove all transitions in an F_j for which $K_i \cap E_j = \emptyset$, and decompose the resulting transition system into its SCCs D_1, \dots, D_l . These are the maximal negative subloops of K_i and for each D_j a word visiting all transitions in D_j is added to S_- . ◀

While every Ω -IRC language can be learned through a characteristic sample, the same does not hold for arbitrary ω -regular languages as the following proposition establishes.



■ **Figure 4** In this figure an excerpt of the transition system for the proof of Proposition 10 is depicted. The transition from q_n to q_j forms a closed loop and words $w_1 \in L_\vee, w_2 \notin L_\vee$ which induce the same infinity set can be found. Based on the existence of these words we can conclude that **Sprout** constructs a chain with self-loops on each state when attempting to learn an automaton recognizing L_\vee .

► **Proposition 10.** *The language $L_\vee = \{w \in \{a, b\}^\omega : aaaa \text{ occurs infinitely often in } w \text{ or } bbbb \text{ occurs infinitely often in } w\}$ cannot be learned by **Sprout**.*

Proof. To simplify notation, we exchange the alphabet and use $\Sigma = \{0, 1\}$ instead, as it allows arithmetic on the symbols in Σ . We prove this claim by showing through induction that the transition system constructed by **Sprout** must be a chain with loops on each state. Specifically we show that every intermediate transition system $\mathcal{T} = (Q, \Sigma, q_0, \delta)$ with $Q = \{q_0, q_1, \dots, q_n\}$ created by **Sprout** before the threshold is exceeded is either not Ω -consistent with L_\vee for any $\Omega \in \text{Acc}$ or the following holds:

- for each $i < n$ there exists a symbol $\sigma \in \Sigma$ such that $\delta(q_i, \sigma) = q_i$ and $\delta(q_i, 1 - \sigma) = q_{i+1}$
- if q_n has an outgoing transition on some $\sigma \in \Sigma$ then $\delta(q_n, \sigma) = q_n$ and $\delta(q_n, 1 - \sigma) = \perp$

The initial transition system is clearly Ω -consistent with L_\vee for all $\Omega \in \text{Acc}$. Further it trivially satisfies the two outlined conditions as it has only one state, for which no outgoing transitions exist. For the induction step assume that **Sprout** has constructed a transition system $\mathcal{T} = (Q, \Sigma, q_0, \delta)$ with $Q = \{q_0, q_1, \dots, q_n\}$ for which the claim holds. We now show that the next inserted transition either introduces an inconsistency with L_\vee or it leads to a transition system that also satisfies the two conditions.

If a transition from q_n to some q_j with $j < n$ were inserted, then a closed cycle is formed. As q_j is reachable there must exist some word $u \in \Sigma^*$ such that $\delta^*(q_0, u) = q_j$. Consider now the word $v \in \Sigma^*$ such that $\delta^*(q_j, v) = q_j$ and the letters in v are such that they alternate between taking the self-loop and moving to the next state along the cycle. If the loop on q_n does not exist, then v just transitions back to q_j at this point. As can be seen in Figure 4, no alphabet symbol can occur more than once in a row in v if the dashed self-loop on q_n is present. Otherwise at most three consecutive occurrences of the same symbol can appear in v and we clearly have that $w_1 = uv^\omega \notin L_\vee$. Consider now a word w_2 which takes each self-loop on the cycle four times before moving to the next state. This means $w_2 \in L_\vee$ but because the infinity sets induced by w_1 and w_2 coincide (as both words take all possible transitions infinitely often), an automaton containing such a closed cycle cannot be consistent with L_\vee .

We have thus shown that no transition can lead from q_n back to a state q_j with $j < n$. If q_n has no outgoing transitions, then a self-loop on the currently escaping symbol is inserted as it clearly does not introduce an inconsistency. On the other hand if q_n already has a self-loop on some symbol $\sigma \in \Sigma$, then the transition on $1 - \sigma$ must lead to a new state q_{n+1} as otherwise $(ab)^\omega \notin L_\vee$ and $(aabb)^\omega \in L_\vee$ would induce the same infinity set. Thus the **Sprout** algorithm indeed constructs a chain with self-loops until it eventually exceeds the threshold. Once this happens, the transition system is extended such that it accepts precisely the positive sample words. As the sample is finite, the resulting automaton cannot recognize L_\vee since there will always be some word $w \in L_\vee$ that is not present in the sample. ◀

However on the other hand **Sprout** is not limited to learning automata for languages with IRC of some type. In the following proposition we give an infinite family of languages which are not in Ω -IRC for any $\Omega \in \text{Acc}$, and have polynomial size characteristic samples for **Sprout**.

► **Proposition 11.** *For $i > 1$, consider $L_i = (\Sigma^*b^i)^\omega$ and the sample $S^i = (S_+^i, S_-^i)$ with $S_+^i = \{b^\omega, (b^i a b^{i-1} a \dots b^2 a b^1 a)^\omega\} \cup \{(b^j a b^k)^\omega : j + k = i\}$ and $S_-^i = \{(b^j a)^\omega : j < i\}$. Then S^i is a characteristic sample for L_i and the learner **Sprout** with parity as target condition. (The sample for $i = 3$ is used in the example in Figure 3.)*

Proof. In the following we show that **Sprout** constructs a DPA for the language L_i from the characteristic sample S^i . Note first that the exit-strings of any two sample words are distinct for every transition system constructed by **Sprout**, since all words in S^i consist of only a periodic part. Further in every word $v^\omega \in S_+^i$ the infix b^i occurs, which means that an infinite run on any positive sample word is only possible in a transition system that permits i consecutive transitions on the symbol b .

Initially, the algorithm inserts a self-loop on a as no sample words prevent this. Subsequently the b -transition cannot be a self-loop as otherwise the infinity sets induced by positive and negative sample words would coincide. Thus a new state is added to which the b -transition from ε leads. We now proceed inductively to show that a b -chain of length $i - 1$ with a -transitions leading back to the initial state is created. We will identify each state on this chain with the minimal word of the form b^j that reaches it.

Formally such a chain satisfies that for all $j < i$ we have $\delta^*(\varepsilon, b^j) = b^j$ and $\delta^*(\varepsilon, b^k a) = \varepsilon$ for all $k < j$. The base case for $j = 1$ has already been described above so assume now that the statement holds for $j - 1$ and consider the two transitions that **Sprout** inserts for the state b^{j-1} . We see that inserting an a -transition from b^{j-1} to ε does not introduce an inconsistency. This is because as outlined above no positive sample word induces an infinite run and the exit string of any two sample words must be distinct.

It remains to be shown that the b -transition from b^{j-1} must lead to a new state b^j . To see this assume to the contrary that the introduction of a b -transition from b^{j-1} to some b^l with $l < j$ leads to a transition system \mathcal{T}' which is Parity-consistent with S^i . It is not hard to see that the infinity set P induced by the positive sample word $(b^i a b^{i-1} \dots b^1 a)^\omega$ contains all transitions in \mathcal{T}' . Now let N_0, N_1, \dots, N_j be the infinity sets induced by the negative sample words $a^\omega, (ba)^\omega, \dots, (b^j a)^\omega$. It is easily verified that $P = N_0 \cup N_1 \cup \dots \cup N_j$, thus satisfying the conditions for Lemma 2. This means that \mathcal{T}' cannot be Parity-consistent with S^i and hence no b -transition from b^{j-1} to any b^l with $l < j$ is kept.

Once this b -chain of length $i - 1$ is constructed, we simply need to verify that inserting both the a - and b -transition from b^{i-1} to ε does not lead to an inconsistent transition system. Since only positive sample words contain i consecutive occurrences of b , the b -transition from b^{i-1} to ε occurs exclusively in the infinity set induced by positive but not negative words. Thus a consistent parity condition exists and **Sprout** constructs a DPA recognizing L_i . ◀

5 Active Learning

We consider the standard minimal adequate teacher (MAT) active learning scenario [1], in which the learning algorithm has access to a teacher that can answer membership queries and equivalence queries for the target language, and returns a counterexample if the automaton for an equivalence query is not correct. A natural extension to ω -automata considers membership queries for ultimately periodic words and equivalence queries with ultimately periodic words as counterexamples (see [16]).

Since there is a polynomial time active learning algorithm for deterministic weak automata [16], a natural next candidate for polynomial time active learning are deterministic automata with an informative right congruence. However, the theorem below basically shows that this class is as hard for active learning as general regular ω -languages.

► **Theorem 12.** *Let $\Omega \in \text{Acc}$ be an acceptance type, and consider the active learning setting with membership and equivalence queries for ultimately periodic words. There is a polynomial time active learning algorithm for deterministic automata of type Ω with informative right congruence if, and only if, there is a polynomial time active learning algorithm for general deterministic automata of type Ω .*

Proof (sketch). Assume that AL_{IRC} is an active learning algorithm for automata with informative right congruence of type Ω . The arguments used below work for all acceptance types $\Omega \in \text{Acc}$. For simplicity we use the parity condition in the following.

Our goal is to use AL_{IRC} in order to define an active learning algorithm AL for general DPA that runs in polynomial time if AL_{IRC} does. The rough idea is as follows: We have to learn an automaton \mathcal{A} for a target language $L \subseteq \Sigma^\omega$ that does not have an IRC, in general. Such an automaton \mathcal{A} can be turned into an automaton with IRC by adding new letters to the alphabet, and then extending the automaton such that from each state a different word over these new letters is accepted. Restricted to the original alphabet, this extended automaton still accepts the same language as before. Since the new automaton has an IRC, we can use AL_{IRC} to learn it. The only problem with this approach is that we do not know the target automaton \mathcal{A} , so we cannot simply extend it and let AL_{IRC} learn the extension. However, we can simulate a teacher for AL_{IRC} that answers queries of AL_{IRC} such that these answers are consistent with such an extension of \mathcal{A} . We give the answers such that they only reveal information on the original target language L . Hence, AL_{IRC} first has to learn, in some sense, an automaton for L in order to obtain information on the newly added letters in the extension.

More formally, define an extended alphabet $\Sigma_\star = \Sigma \dot{\cup} \{\star, 0, 1\}$ with new letters $\star, 0, 1$ that do not occur in Σ . Now let $L \subseteq \Sigma^\omega$ be a target language which we want to learn. Our algorithm AL simulates AL_{IRC} over the alphabet Σ_\star . Note that AL has access to a teacher T that answers queries for the language L over the alphabet Σ . We define a teacher T_{IRC} that answers queries that are asked by AL_{IRC} during its simulation as follows:

- *Membership query for a word $w = uv^\omega$:* If none of the newly introduced symbols occur in w , i.e. $w \in \Sigma^\omega$ then we simply copy the answer $T(w)$. Otherwise w must contain $0, 1$ or \star in which case T_{IRC} always gives a negative answer.
- *Equivalence query for an automaton \mathcal{A} :* We construct a new automaton \mathcal{B} by removing from \mathcal{A} all transitions on symbols $0, 1$ or \star and pruning any unreachable states. \mathcal{B} is then given to T for an equivalence query. If $T(\mathcal{B})$ returns a counterexample w , then this is used as the result of $T_{IRC}(\mathcal{A})$.

Otherwise the automaton \mathcal{B} must recognize the target language L . In this case, the simulation of AL_{IRC} is stopped, and our algorithm AL returns \mathcal{B} .

It can be shown that this algorithm AL learns the target language L in polynomial time if AL_{IRC} is a polynomial time algorithm. ◀

So the property of an IRC does not help for active learning, while for passive learning in the limit it seems to make the problem simpler. We finish this section with the observation that polynomial time active learning is at least as hard as learning in the limit with polynomial time and data, given that the class \mathcal{K} of target automata satisfies the following properties (which are satisfied by standard classes of deterministic automata):

20:16 Constructing Deterministic ω -Automata from Examples

- (P1) It is decidable in polynomial time if a given word is accepted by a given automaton from \mathcal{K} .
- (P2) For a given sample S , one can construct in polynomial time an automaton from \mathcal{K} that is consistent with S .
- (P3) If two automata from \mathcal{K} are not equivalent, then there exists a word of polynomial size witnessing the difference.

► **Proposition 13.** *Consider a class \mathcal{K} of finite automata for which properties (P1)–(P3) are satisfied. If there is a polynomial time active learning algorithm for \mathcal{K} , then \mathcal{K} can be learned in the limit with polynomial time and data.*

Proof (sketch). Assume that there is a polynomial time active learning algorithm $AL_{\mathcal{K}}$ for target automata from \mathcal{K} . A passive learner can simulate an execution of $AL_{\mathcal{K}}$ in which equivalence queries are always answered with the smallest counterexample. A characteristic sample can be constructed from all the words that are used in such an execution of $AL_{\mathcal{K}}$. ◀

6 Conclusion

We have presented polynomial time algorithms for checking the consistency of a (partial) deterministic transition system with a set of positive and negative ultimately periodic words for the acceptance conditions Büchi, generalized Büchi, parity, and Rabin. Since co-Büchi and Streett conditions are dual to Büchi and Rabin conditions, respectively, one also obtains algorithms for these conditions by flipping negative and positive examples.

The consistency algorithms allow us to extend the principle of the RPNI algorithm from finite to infinite words, leading to the polynomial time algorithm **Sprout** that constructs a deterministic ω -automaton from given ultimately periodic examples. We have shown that **Sprout** can learn deterministic automata for languages with an IRC in the limit with polynomial time and data. While **Sprout** is not restricted to IRC languages, there are regular ω -languages which it cannot learn. It is obviously an interesting open question whether there is an algorithm that learns deterministic automata for general regular ω -languages with polynomial time and data. Our results in Section 5 show that finding such an algorithm is not more difficult than finding an active learning algorithm that learns deterministic automata for IRC languages from membership and equivalence queries.

References

- 1 Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987. doi:10.1016/0890-5401(87)90052-6.
- 2 Dana Angluin and Dana Fisman. Learning regular omega languages. *Theor. Comput. Sci.*, 650:57–72, 2016. doi:10.1016/j.tcs.2016.07.031.
- 3 Dana Angluin and Dana Fisman. Regular omega-languages with an informative right congruence. In *Proceedings Ninth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2018, Saarbrücken, Germany, 26-28th September 2018*, volume 277 of *EPTCS*, pages 265–279, 2018. doi:10.4204/EPTCS.277.19.
- 4 Dana Angluin, Dana Fisman, and Yaara Shoval. Polynomial identification of ω -automata. In Armin Biere and David Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part II*, volume 12079 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2020. doi:10.1007/978-3-030-45237-7_20.

- 5 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
- 6 Andreas Birkendorf, Andreas Böker, and Hans Simon. Learning deterministic finite automata from smallest counterexamples. *SIAM J. Discrete Math.*, 13:465–491, January 2000. doi:10.1137/S0895480198340943.
- 7 J Richard Büchi. On a decision method in restricted second order arithmetic, logic, methodology and philosophy of science (proc. 1960 internat. congr.), 1962.
- 8 Rafael C. Carrasco and José Oncina. Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21-23, 1994, Proceedings*, volume 862 of *Lecture Notes in Computer Science*, pages 139–152. Springer, 1994. doi:10.1007/3-540-58473-0_144.
- 9 Olivier Carton and Ramón Maceiras. Computing the rabin index of a parity automaton. *RAIRO – Theoretical Informatics and Applications – Informatique Théorique et Applications*, 33(6):495–505, 1999. URL: http://www.numdam.org/item/ITA_1999__33_6_495_0/.
- 10 Colin De La Higuera. Characteristic sets for polynomial grammatical inference. In Laurent Miclet and Colin de la Higuera, editors, *Grammatical Interference: Learning Syntax from Sentences*, pages 59–71, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.
- 11 Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science, LICS '97*, pages 99–110, Los Alamitos, California, 1997. IEEE Computer Society Press. doi:10.1109/lics.1997.614939.
- 12 E. Mark Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978. doi:10.1016/S0019-9958(78)90562-4.
- 13 John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and their Relation to Automata*. Addison-Wesley, 1969.
- 14 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. doi:10.1007/978-1-4684-2001-2_9.
- 15 Christof Löding and Anton Pirogov. Determinization of Büchi automata: Unifying the approaches of Safra and Muller-Schupp. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, volume 132 of *LIPICs*, pages 120:1–120:13. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPICs.ICALP.2019.120.
- 16 Oded Maler and Amir Pnueli. On the learnability of infinitary regular sets. *Inf. Comput.*, 118(2):316–326, 1995. doi:10.1006/inco.1995.1070.
- 17 Hua Mao, Yingke Chen, Manfred Jaeger, Thomas D. Nielsen, Kim G. Larsen, and Brian Nielsen. Learning probabilistic automata for model checking. In *Eighth International Conference on Quantitative Evaluation of Systems, QEST 2011, Aachen, Germany, 5-8 September, 2011*, pages 111–120. IEEE Computer Society, 2011. doi:10.1109/QEST.2011.21.
- 18 Philipp J. Meyer, Salomon Sickert, and Michael Luttenberger. Strix: Explicit reactive synthesis strikes back! In *Computer Aided Verification – 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, pages 578–586, 2018. doi:10.1007/978-3-319-96145-3_31.
- 19 Jakub Michaliszyn and Jan Otop. Learning deterministic automata on infinite words. In *ECAI 2020 – 24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2370–2377. IOS Press, 2020. doi:10.3233/FAIA200367.
- 20 José Oncina, Pedro García, and Enrique Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):448–458, 1993. doi:10.1109/34.211465.
- 21 Jose Oncina and Pedro García. Inferring regular languages in polynomial update time. *World Scientific*, January 1992. doi:10.1142/9789812797902_0004.
- 22 Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proceedings of the 21st IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 255–264. IEEE Computer Society, 2006. doi:10.2168/LMCS-3(3:5)2007.

- 23 Shmuel Safra. On the complexity of omega-automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FoCS '88*, pages 319–327, Los Alamitos, California, 1988. IEEE Computer Society Press. doi:10.1109/SFCS.1988.21948.
- 24 Sven Schewe. Tighter bounds for the determinisation of Büchi automata. In *Proceedings of Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009*, volume 5504 of *Lecture Notes in Computer Science*, pages 167–181. Springer, 2009. doi:10.1007/978-3-642-00596-1_13.
- 25 Sven Schewe. Beyond Hyper-Minimisation – Minimising DBAs and DPAs is NP-Complete. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.FSTTCS.2010.400.
- 26 Wolfgang Thomas. *Automata on Infinite Objects*, page 133–191. MIT Press, Cambridge, MA, USA, 1991.
- 27 Wolfgang Thomas. Facets of synthesis: Revisiting Church’s problem. In *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures, FOSSACS 2009*, volume 5504 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2009. doi:10.1007/978-3-642-00596-1_1.
- 28 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.



Computational Complexity of Covering Multigraphs with Semi-Edges: Small Cases

Jan Bok  

Computer Science Institute, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Jiří Fiala  

Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Petr Hliněný  

Faculty of Informatics, Masaryk University, Brno, Czech Republic

Nikola Jedličková  

Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Jan Kratochvíl  

Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Abstract

We initiate the study of computational complexity of graph coverings, aka locally bijective graph homomorphisms, for *graphs with semi-edges*. The notion of graph covering is a discretization of coverings between surfaces or topological spaces, a notion well known and deeply studied in classical topology. Graph covers have found applications in discrete mathematics for constructing highly symmetric graphs, and in computer science in the theory of local computations. In 1991, Abello et al. asked for a classification of the computational complexity of deciding if an input graph covers a fixed target graph, in the ordinary setting (of graphs with only edges). Although many general results are known, the full classification is still open. In spite of that, we propose to study the more general case of covering graphs composed of normal edges (including multiedges and loops) and so-called semi-edges. Semi-edges are becoming increasingly popular in modern topological graph theory, as well as in mathematical physics. They also naturally occur in the local computation setting, since they are lifted to matchings in the covering graph. We show that the presence of semi-edges makes the covering problem considerably harder; e.g., it is no longer sufficient to specify the vertex mapping induced by the covering, but one necessarily has to deal with the edge mapping as well. We show some solvable cases and, in particular, completely characterize the complexity of the already very nontrivial problem of covering one- and two-vertex (multi)graphs with semi-edges. Our NP-hardness results are proven for simple input graphs, and in the case of regular two-vertex target graphs, even for bipartite ones. We remark that our new characterization results also strengthen previously known results for covering graphs without semi-edges, and they in turn apply to an infinite class of simple target graphs with at most two vertices of degree more than two. Some of the results are moreover proven in a more general setting (e.g., finding k -tuples of pairwise disjoint perfect matchings in regular graphs, or finding equitable partitions of regular bipartite graphs).

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases graph cover, covering projection, semi-edges, multigraphs, complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.21

Related Version The full proofs of *-marked statements can be found in the extended version of this paper.

Extended Version: <https://arxiv.org/abs/2103.15214>



© Jan Bok, Jiří Fiala, Petr Hliněný, Nikola Jedličková, and Jan Kratochvíl;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 21; pp. 21:1–21:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Jan Bok*: Supported by research grant GAČR 20-15576S of the Czech Science Foundation and by SVV-2020-260578. The author was also partially supported by GAUK 1580119.

Jiří Fiala: Supported by research grant GAČR 20-15576S of the Czech Science Foundation.

Petr Hliněný: Supported by research grant GAČR 20-04567S of the Czech Science Foundation.

Nikola Jedličková: Supported by research grant GAČR 20-15576S of the Czech Science Foundation and by SVV-2020-260578.

Jan Kratochvíl: Supported by research grant GAČR 20-15576S of the Czech Science Foundation.

1 Introduction

1.1 Graph coverings and complexity

The notion of a *graph covering* is a discretization of coverings between surfaces or topological spaces, a notion well known and deeply studied in classical topology. Graph coverings have found many applications. Primarily as a tool for construction of highly symmetric graphs [5, 15, 24, 27], or for embedding complete graphs in surfaces of higher genus [48].

Graph coverings attracted attention of computer scientists as well. Angluin [2] exploited graph covers when introducing models of local computations, namely by showing that a graph and its cover cannot be distinguished by local computations. Later, Litovsky et al. [39] proved that planar graphs and series-parallel graphs cannot be recognized by local computations, and Courcelle and Metivier [14] showed that in fact no nontrivial minor-closed class of graphs can. In both of these results, graph coverings were used as the main tool, as well as in more recent papers of Chalopin et al. [8, 9]. Here, the authors presented a model for distributed computations and addressed the algorithmic complexity of problems associated with such a model. To this end, they used the existing results on NP-completeness of the covering problem to provide their hardness results. In [10], the authors study a close relation of packing bipartite graphs to a special variant of graph coverings called *pseudo-coverings*.

Another connection to algorithmic theory comes through the notions of the *degree partition* and the *degree refinement matrix* of a graph. These notions were introduced by Corneill [12, 13] in hope of solving the graph isomorphism problem efficiently. It can be easily seen that a graph and all of its covers have the same degree refinement matrix. Motivated by this observation, Angluin and Gardiner [3] proved that any two finite regular graphs of the same valency have a finite common cover, and conjectured the same for every two finite graphs with the same degree refinement matrix, which was proved by Leighton [37].

The stress on finiteness of the common cover is natural. For every matrix, there exists a universal cover, an infinite tree, that covers all graphs with this degree refinement matrix. Trees are planar graphs, and this inspired an at first sight innocent question of which graphs allow a finite planar cover. Negami observed that projective planar graphs do (in fact, their double planar covers characterize their projective embedding), and conjectured that these two classes actually coincide [46]. Despite a serious effort of numerous authors, the problem is still open, although the scope for possible failure of Negami's conjecture has been significantly reduced [4, 28, 29].

A natural computational complexity question is how difficult is to decide, given two graphs, if one covers the other one. This question is obviously at least as difficult as the graph isomorphism problem (consider two given graphs on the same number of vertices). It was proven NP-complete by Bodlaender [7] (in the case of both graphs being part of the input). Abello et al. [1] initiated the study of the computational complexity of the H -cover problem for a fixed target graph H by showing that deciding if an input graph covers the dumbbell graph $W(0, 1, 1, 1, 0)$ (in our notation from Section 4) is NP-complete (note that the dumbbell

graph has loops, and they also allowed the input graph to contain loops). Furthermore, they asked for a complete characterization of the computational complexity, depending on the parameter graphs H . Such a line of research was picked by Kratochvíl, Proskurowski and Telle, who completely characterized the complexity for simple target graphs with at most 6 vertices [33], and then noted that in order to fully characterize the complexity of the H -cover problem for simple target graphs, it is sufficient (but also necessary) to classify it for mixed colored multigraphs with minimum degree at least three [30]. The latter result gives a hope for a more concise description of the characterization, but is also in line with the original motivation from topological graph theory, where loops and multiedges are widely considered.

The complexity of covering 2-vertex multigraphs was fully characterized in [30], the characterization for 3-vertex undirected multigraphs can be found in [34]. The most general NP-hardness result known so far is the hardness of covering simple regular graphs of valency at least three [32, 17]. More recently, Bílka et al. [6] proved that covering several concrete small graphs (including the complete graphs K_4 , K_5 and K_6) remains NP-hard for planar inputs. This shows that planarity does not help in graph covering problems in general, yet the conjecture that the H -COVER problem restricted to planar inputs is at least as difficult as for general inputs, provided H itself has a finite planar cover, remains still open. Planar graphs have also been considered by Fiala et al. [19] who showed that for planar input graphs, H -REGULARCOVER is in FPT when parameterized by H . This is in fact the first and only paper on the complexity of regular covers, i.e., covering projections determined by a regular action of a group of automorphisms on the covering graph.

Graph coverings were also extensively studied under a unifying umbrella of *locally constrained homomorphisms*. In these relaxations, homomorphisms can be either locally injective or locally surjective and not necessarily locally bijective. The computational complexity of locally surjective homomorphisms has been classified completely, with respect to the fixed target graph [22]. Though the complete classification of the complexity of locally injective homomorphisms is still out of sight, it has been proved for its list variant [16]. The problem is also interesting for its applied motivation – a locally injective homomorphism into the complement of a path of length k corresponds to an $L(2, 1)$ -labeling of span k , an intensively studied notion stemming from the theory of frequency assignment. Further generalizations include the notion of $H(p, q)$ -coloring, a homomorphism into a fixed target graph H with additional rules on the neighborhoods of the vertices [18, 35]. To find more about locally injective homomorphisms, see e.g. [41, 11] or [21]. For every fixed graph H , the existence of a locally injective homomorphism to H is provably at least as hard as the H -cover problem. In this sense our hardness results extend the state of the art also for the problem of existence of locally injective homomorphisms.

1.2 Graphs with semi-edges

The notion of *semi-edges* has been introduced in the modern topological graph theory and it is becoming more and more frequently used (the terminology has not yet stabilized; semi-edges are often called half-edges, and sometimes fins). Mednykh and Nedela recently wrote a monograph [44] in which they summarize and survey the ambitions and efforts behind generalizing the notion of graph coverings to the graphs with semi-edges. This generalization, as the authors pinpoint, is not artificial as such graphs emerge “in the situation of taking quotients of simple graphs by groups of automorphisms which are semiregular on vertices and darts (arcs) and which may fix edges”. As the authors put it: “A problem arises when one wants to consider quotients of such graphs (graphs embedded to surfaces) by an involution fixing an edge e but transposing the two incident vertices. The edge e is halved and mapped

to a semi-edge – an edge with one free end.” This direction of research proved to be very fruitful and provided many applications and generalizations to various parts of algebraic graph theory. For example, Malnič et al. [42] considered semi-edges during their study of abelian covers and as they write “...in order to have a broader range of applications we allow graphs to have semi-edges.” To highlight a few other contributions, the reader is invited to consult [45, 43], the surveys [36] and (aforementioned) [44], and finally for more recent results the series of papers [19, 23, 20]. It is also worth noting that the concept of graphs with semi-edges was introduced independently and naturally in mathematical physics by Getzler and Karpanov [26].

In the view of the theory of local computations, semi-edges and their covers prove very natural, too, and it is even surprising that they have not been considered before in the context. If a computer network is constructed as a cover of a small template, the preimages of normal edges in the covering projection are matchings completely connecting nodes of two types (the end-vertices of the covered edge). Preimages of loops are disjoint cycles with nodes of the same type. And preimages of semi-edges are matchings on vertices of the same type. The role of semi-edges was spotted by Woodhouse et. al. [51, 49] who have generalized the fundamental theorem of Leighton on finite common covers of graphs with the same degree refinement matrix to graphs with semi-edges.

Our goal is to initiate the study of the computational complexity of covering graphs with semi-edges, and the current paper is opening the door in this direction.

1.3 Formal definitions

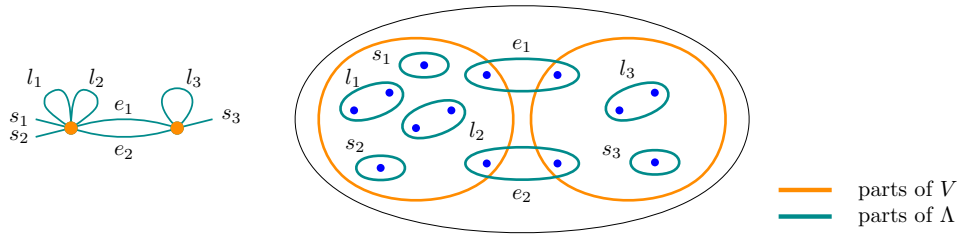
In this subsection we formally define what we call *graphs*. A graph has a set of vertices and a set of edges (also referred to as links). As it is standard in topological graph theory, we automatically allow multiple edges and loops. Every ordinary edge is connecting two vertices, every loop is incident with only one vertex. On top of these, we also allow *semi-edges*. Each semi-edge is also incident with only one vertex. The difference between loops and semi-edges is that a loop contributes two to the degree of its vertex, while a semi-edge only one. A very elegant description of ordinary edges, loops and semi-edges through the concept of *darts* is used in more algebraic-based papers on covers. The following formal definition is a reformulation of the one given in [44].

► **Definition 1.** A graph is a triple (D, V, Λ) , where D is a set of darts, and V and Λ are each a partition of D into disjoint sets. Moreover, all sets in Λ have size one or two.

With this definition, the *vertices* of a graph (D, V, Λ) are the sets of V (note that empty sets correspond to isolated vertices, and since we are interested in covers of connected graphs by connected ones, we assume that all sets of V are nonempty). The sets of Λ are referred to as *links*, and they are of three types – loops (2-element sets with both darts from the same set of V), (ordinary) edges (2-element sets intersecting two different sets of V), and semi-edges (1-element sets). After this explanation it should be clear that this definition is equivalent to a definition of multigraphs which is standard in the graph theory community:

► **Definition 2.** A graph is an ordered triple (V, Λ, ι) , for $\Lambda = E \cup L \cup S$, where ι is the incidence mapping $\iota : \Lambda \rightarrow V \cup \binom{V}{2}$ such that $\iota(e) \in V$ for all $e \in L \cup S$ and $\iota(e) \in \binom{V}{2}$ for all $s \in E$.

For a comparison of Definitions 1 and 2, see Figure 1. Since we consider multiple edges of the same type incident with the same vertex (or with the same pair of vertices), the edges are given by their names and the incidence mapping ι expresses which vertex (or vertices) “belong” to a particular edge. The degree of a vertex is then defined as follows.



■ **Figure 1** An example of a graph presented in a usual graph-theoretical way (left) and using the dart-based Definition 1 (right).

► **Definition 3.** For a graph $G = (V, \Lambda = E \cup L \cup S, \iota)$, the degree of a vertex $u \in V$ is defined as

$$\deg_G(u) = p_S(u) + p_E(u) + 2p_L(u),$$

where $p_S(u)$ ($p_L(u)$) is the number of semi-edges $e \in S$ (of loops $e \in L$) such that $\iota(e) = u$, and $p_E(u)$ is the number of ordinary edges $e \in E$ such that $u \in \iota(e)$.

We call a graph G *simple* if $p_S(u) = p_L(u) = 0$ for every vertex $u \in V(G)$ (the graph has no loops or semi-edges) and $\iota(e) \neq \iota(e')$ for every two distinct $e, e' \in E$ (the graph has no multiple (ordinary) edges). We call G *semi-simple* if $p_S(u) \leq 1$ and $p_L(u) = 0$ for every vertex $u \in V(G)$ and $\iota(e) \neq \iota(e')$ for every two distinct $e, e' \in E$.

Note that in the language of Definition 1, the degree of a vertex $v \in V$ is simply $|v|$. And in this language, the main object of our study, a *graph cover* (or equivalently a *covering projection*), is defined as follows.

► **Definition 4.** We say that a graph $G = (D_G, V_G, \Lambda_G)$ covers a connected graph $H = (D_H, V_H, \Lambda_H)$ (denoted as $G \rightarrow H$) if there exists a map $f : D_G \rightarrow D_H$ such that:

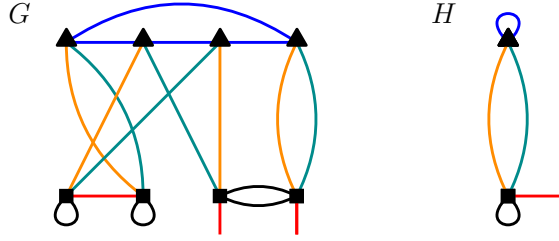
- For every $u \in V_G$, there is a $u' \in V_H$ such that the restriction of f onto u is a bijection between u and u' .
- For every $e \in \Lambda_G$, there is an $e' \in \Lambda_H$ such that $f(e) = e'$.

The map f is called *graph cover* (or *covering projection*).

One must appreciate how compact and elegant this definition is after translating it into the language of Definition 2 in Proposition 5, which otherwise is the definition of (multi)graph covering in the standard language of Definition 2.

► **Proposition 5.** A graph G covers a graph H if and only if G allows a pair of mappings $f_V : V(G) \rightarrow V(H)$ and $f_\Lambda : \Lambda(G) \rightarrow \Lambda(H)$ such that

1. $f_\Lambda(e) \in L(H)$ for every $e \in L(G)$ and $f_\Lambda(e) \in S(H)$ for every $e \in S(G)$,
2. $\iota(f_\Lambda(e)) = f_V(\iota(e))$ for every $e \in L(G) \cup S(G)$,
3. for every link $e \in \Lambda(G)$ such that $f_\Lambda(e) \in S(H) \cup L(H)$ and $\iota(e) = \{u, v\}$, we have $\iota(f_\Lambda(e)) = f_V(u) = f_V(v)$,
4. for every link $e \in \Lambda(G)$ such that $f_\Lambda(e) \in E(H)$ and $\iota(e) = \{u, v\}$ (note that it must be $f_V(u) \neq f_V(v)$), we have $\iota(f_\Lambda(e)) = \{f_V(u), f_V(v)\}$,
5. for every loop $e \in L(H)$, $f^{-1}(e)$ is a disjoint union of loops and cycles spanning all vertices $u \in V(G)$ such that $f_V(u) = \iota(e)$,
6. for every semi-edge $e \in S(H)$, $f^{-1}(e)$ is a disjoint union of edges and semi-edges spanning all vertices $u \in V(G)$ such that $f_V(u) = \iota(e)$, and
7. for every edge $e \in E(H)$, $f^{-1}(e)$ is a disjoint union of edges (i.e., a matching) spanning all vertices $u \in V(G)$ such that $f_V(u) \in \iota(e)$.



■ **Figure 2** An example of a covering. The vertex mapping of the covering from G to H is determined by the shape of the vertices, the edge mapping by the colors of the edges.

See an example of a covering projection in Fig. 2. Conditions 1–4. express the fact that f_V and f_E commute with ι , i.e., that f is a homomorphism from G to H . Conditions 5–7 express that this homomorphism is locally bijective (for every ordinary edge e incident with $f_V(u)$ in H , there is exactly one ordinary edge of G which is incident with u and mapped to e by f_E ; for every semi-edge e incident to $f_V(u)$ in H , there is exactly one semi-edge, or exactly one ordinary edge (but not both) in G incident with u and mapped to e by f_E ; and for every loop e incident with $f_V(u)$ in H , there is exactly one loop or exactly two ordinary edges (but not both) of G which are incident with u and mapped to e by f_E).

Even though the aforementioned definitions of graphs and graph covers through darts are compact and elegant, in the rest of the paper we shall work with the standard definition of graphs and the equivalent description of graph covers given by Proposition 5, because they are better suited for describing the reductions and understanding the illustrative figures.

It is clear that a covering projection (more precisely, its vertex mapping) preserves degrees. One may ask when (or if) a degree preserving vertex mapping can be extended to a covering projection. An obvious necessary condition is described by the following definition.

► **Definition 6.** A vertex mapping $f_V : V(G) \rightarrow V(H)$ between graphs G and H is called degree-obedient if

1. for any two distinct vertices $u, v \in V(H)$ and any vertex $x \in f_V^{-1}(u)$, the number of ordinary edges e of H such that $\iota(e) = \{u, v\}$ equals the number of ordinary edges of G with one end-vertex x and the other one in $f_V^{-1}(v)$, and
2. for every vertex $u \in V(H)$ and any vertex $x \in f_V^{-1}(u)$, the value $p_{S(H)}(u) + 2p_{L(H)}(u)$ equals $p_{S(G)}(x) + 2p_{L(G)}(x) + r$, where r is the number of edges of G with one end-vertex x and the other one from $f_V^{-1}(u) \setminus \{x\}$,
3. for every vertex $u \in V(H)$ and any vertex $x \in f_V^{-1}(u)$, $p_{S(G)}(x) \leq p_{S(H)}(u)$.

Finally, let us recall that the product $G \times H$ of graphs G and H is defined as the graph with the vertex set being the Cartesian product $V(G) \times V(H)$ and with vertices (u, v) and (u', v') being adjacent in $G \times H$ if and only if u is adjacent to u' , and v is adjacent to v' .

1.4 Overview of our results

The first major difference between graphs with and without semi-edges is that for target graphs without semi-edges, every degree-obedient vertex mapping to it can be extended to a covering. This is not true anymore when semi-edges are allowed (consider a one-vertex graph with three semi-edges, every 3-regular graph allows a degree-obedient mapping onto it, but only the 3-edge-colorable ones are covering it). In Section 2 we show that the situation is not as bad if the source graph is bipartite. In Theorem 10 we prove that if the source graph is

bipartite and has no semi-edges, then every degree-obedient vertex mapping can be extended to a covering, while if semi-edges are allowed in the bipartite source graph, it can at least be decided in polynomial time if a degree-obedient mapping can be extended to a covering.

All other results concern the complexity of the following decision problem

PROBLEM: H -COVER
 INPUT: A graph G .
 QUESTION: Does G cover H ?

In order to present our results in the strongest possible form, we aim at proving the hardness results for restricted classes of input graphs, while the polynomial ones for the most general inputs. In particular, we only allow simple graphs as inputs when we prove NP-hardness, and on the other hand, we allow loops, multiple edges as well as semi-edges when we present polynomial-time algorithms.

The first NP-hardness result is proven in Theorem 11, namely that covering semi-simple regular graphs of valency at least 3 is NP-hard even for simple bipartite input graphs. In Sections 3 and 4 we give a complete classification of the computational complexity of covering graphs with one and two vertices. This extends the main result of [30] to graphs with semi-edges. Moreover, we strengthen the hardness results of [30] considerably by showing that all NP-hard cases of covering regular two-vertex graphs (even those without semi-edges) remain NP-hard for simple *bipartite* input graphs. It must be noted that through the reduction from [31], our results on the complexity of covering one- or two-vertex graphs provide characterization results on infinitely many simple graphs which contain at most two vertices of degrees greater than 2.

All considered computational problems are clearly in the class NP, and thus we only concentrate on the NP-hardness proofs in the NP-completeness results. We restrict our attention to connected target graphs, in which case it suffices to consider only connected input graphs. In this case every cover is a k -fold cover for some k , which means that the preimage of every vertex has the same size.

2 The impact of semi-edges

In this section we demonstrate the huge difference between covering graphs with and without semi-edges. First, we discuss the necessity of specifying the edge mapping in a covering projection. In other words, we discuss when a degree mapping can always be extended to a covering, and when this question can be decided efficiently. The following proposition follows straightforwardly from the definitions.

► **Proposition 7.** *For every graph covering projection between two graphs, the vertex mapping induced by this projection is degree-obedient.*

► **Proposition * 8.** *If H has no semi-edges, then for any graph G , any degree-obedient mapping from the vertex set of G onto the vertex set of H can be extended to a graph covering projection of G to H .*

Proof sketch. For simple graphs G , this is proved already in [30]. If multiple edges and loops are allowed, we use a similar approach. The key point is that Petersen theorem [47] about 2-factorization of regular graphs of even valence is true for multigraphs without semi-edges as well, and the same holds true for König-Hall theorem [40] on 1-factorization of regular bipartite multigraphs. ◀

As we will see soon, the presence of semi-edges changes the situation a lot. Even for simple graphs, degree-obedient vertex mappings to a graph with semi-edges may not extend to a graph covering projection, and the possibility of such an extension may even be NP-complete.

► **Observation 9.** *Let $F(3,0)$ be the graph with one vertex and three semi-edges pending on this vertex. Then a graph covers $F(3,0)$ if and only if it is 3-regular and 3-edge-colorable. Testing 3-edge-colorability is well known to be NP-hard even for simple graphs.*

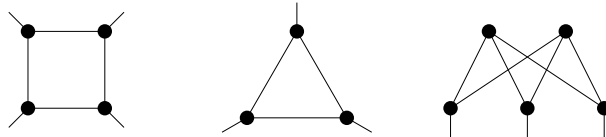
However, if the input graph is bipartite, the situation gets much easier.

► **Theorem * 10.** *If a graph G is bipartite, then for any graph H , it can be decided in polynomial time whether a degree-obedient mapping from the vertex set of G onto the vertex set of H can be extended to a graph covering projection of G to H . In particular, if G has no semi-edges and is bipartite, then every degree-obedient mapping from the vertex set of G onto the vertex set of H can be extended to a graph covering projection of G to H .*

Proof sketch. To prove this statement, it is enough to analyze the edges of H and their preimages in G according to the following classification:

- For each vertex pair $x \neq y \in V(H)$ inducing $k \geq 0$ parallel edges in H , their preimage forms a k -regular subgraph $G_{x,y}$ of bipartite G , and hence $G_{x,y}$ is k -edge colorable which immediately gives a covering projection for these edges.
- For each vertex $x \in V(H)$ with $b \geq 0$ semi-edges and $c \geq 0$ loops incident to x in H , these semi-edges and loops lift to a $(b + 2c)$ -regular subgraph \widetilde{G}_x of G . The algorithmic task now is to decide whether \widetilde{G}_x admits a factor projecting onto the semi-edges incident to x (this is efficiently solvable, e.g., by network flows since \widetilde{G}_x is again bipartite). If the answer is true, a projection of the remaining edges onto the loops incident to x always exists by Petersen theorem. ◀

Now we prove the first general hardness result, namely that covering semi-simple regular graphs is always NP-complete (this is the case when every vertex of the target graph is incident with at most one semi-edge, and the graph has no multiple edges nor loops). See Fig. 3 for examples of semi-simple graphs H defining such hard cases.



■ **Figure 3** Examples of small semi-simple graphs which define NP-complete covering problems.

► **Theorem 11.** *Let H be a semi-simple k -regular graph, with $k \geq 3$. Then the H -COVER problem is NP-complete even for simple bipartite input graphs.*

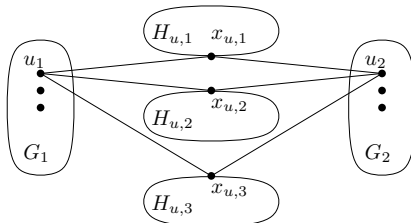
Proof. Consider $H' = H \times K_2$. This graph is simple, k -regular and bipartite, hence the H' -COVER problem is NP-complete by [32]. Given an input k -regular graph G , it is easy to see that G covers H' if and only if it is bipartite and covers H . Since bipartiteness can be checked in polynomial time, the claim follows. ◀

3 One-vertex target graphs

We start the section by proving a slightly more general hardness result, which may be of interest on its own. In particular, it implies that for every $d \geq 3$, it is NP-complete to decide if a simple d -regular graph contains an even 2-factor, i.e., a spanning 2-regular subgraph whose every cycle has even length.

► **Theorem * 12.** *For every $k \geq 2$ and every $d \geq k + 1$, it is NP-complete to decide if a simple d -regular graph contains k pairwise disjoint perfect matchings.*

Proof sketch. The complement of the union of k pairwise disjoint perfect matchings in a $(k + 1)$ -regular graph is a perfect matching as well, and thus a $(k + 1)$ -regular graph contains k pairwise disjoint perfect matchings if and only if it is $(k + 1)$ -edge colorable. Hence for $d = k + 1$, the claim follows from the NP-completeness of d -edge colorability of d -regular graphs which has been proven by Leven and Galil [38].



■ **Figure 4** An illustration to the construction of the graph G' in the proof of Theorem 12.

For $d \geq k + 2$, we reduce from the previous case, as we sketch next. If G is a $(k + 1)$ -regular instance (of the k disjoint perfect matchings problem), we construct an equivalent d -regular instance G' starting from two copies G_1 and G_2 of G , as shown in Fig. 4. Then for each vertex u of G , we connect its two copies u_1, u_2 (in G_1, G_2) by $d - k - 1$ paths of length 2, and add copies of a suitable gadget H to the middle vertices of those paths. The purpose of this gadget is two-fold – it raises all degrees to d , and it prevents edges of the incident path from being used in a perfect matching since the gadget is of an even order. It is routine to finish the construction and to show that G contains k disjoint perfect matchings if and only if G' does so. ◀

Now we are ready to prove a dichotomy theorem on the complexity of covering one-vertex graphs. Let us denote by $F(b, c)$ the one-vertex graph with b semi-edges and c loops.

► **Theorem * 13.** *The $F(b, c)$ -COVER problem is polynomial-time solvable if $b \leq 1$, or $b = 2$ and $c = 0$, and it is NP-complete otherwise, even for simple graphs.*

Proof sketch. The high-level idea is similar to the second point of the proof of Theorem 10: The input graph G possibly covering $F(b, c)$ should better be $(b + 2c)$ -regular (which can be easily checked), and it remains to argue that such G covers $F(b, c)$ if and only if it contains b pairwise disjoint perfect matchings (then a covering projection onto the c loops follows easily). The cases of $b = 0$, $b = 1$, or $b = 2$ and $c = 0$ can be efficiently solved using standard tools, while the remaining cases are hard from Theorem 12 by setting $k = b$ and $d = b + 2c$. ◀

4 Two-vertex target graphs

Let $W(k, m, \ell, p, q)$ be the two-vertex graph with k semi-edges and m loops at one vertex, p loops and q semi-edges at the other one, and $\ell > 0$ multiple edges connecting the two vertices (these edges are referred to as *bars*). In other words, $W(k, m, \ell, p, q)$ is obtained from the disjoint union of $F(k, m)$ and $F(q, p)$ by connecting their vertices by ℓ parallel edges. For an example see the graph H from Fig. 2 which is isomorphic to both $W(1, 1, 2, 1, 0)$ and $W(0, 1, 2, 1, 1)$.

► **Theorem 14.** *The $W(k, m, \ell, p, q)$ -COVER problem is solvable in polynomial time in the following cases*

1. $k + 2m \neq 2p + q$ and ($k \leq 1$ or $k = 2$ and $m = 0$) and ($q \leq 1$ or $q = 2$ and $p = 0$)
2. $k + 2m = 2p + q$ and $\ell = 1$ and $k = q \leq 1$ and $m = p = 0$
3. $k + 2m = 2p + q$ and $\ell > 1$ and $k = m = p = q = 0$

and it is NP-complete otherwise.

Note that case 1 applies to non-regular target graph W , while cases 2 and 3 apply to regular graphs W , i.e., they cover all cases when $k + 2m + \ell = 2p + q + \ell$.

We will refer to the vertex with k semi-edges as *blue* and the vertex with q semi-edges as *red*. In a covering projection $f = (f_V, f_E)$ from a graph G onto $W(k, m, \ell, p, q)$, we view the restricted vertex mapping f_V as a coloring of $V(G)$. We call a vertex $u \in V(G)$ blue (red) if f_V maps u onto the blue (red, respectively) vertex of $W(k, m, \ell, p, q)$. In order to keep the text clear and understandable, we divide the proof into a sequence of claims in separate subsections. This will also allow us to state several hardness results in a stronger form.

4.1 Polynomial parts of Theorem 14

We follow the case-distinction from the statement of Theorem 14:

1. If $k + 2m \neq 2p + q$, then the two vertex degrees of $W(k, m, \ell, p, q)$ are different, and the vertex restricted mapping is uniquely defined for any possible graph covering projection from the input graph G to $W(k, m, \ell, p, q)$. For this coloring of G , if it exists, we check if it is degree-obedient. If not, then G does not cover $W(k, m, \ell, p, q)$. If yes, we check using Theorem 12 whether the blue subgraph of G covers $F(k, m)$ and whether the red subgraph of G covers $F(q, p)$. If any one of them does not, then G does not cover $W(k, m, \ell, p, q)$. If both of them do, then G covers $W(k, m, \ell, p, q)$, since the “remaining” subgraph of G formed by edges with one end-vertex red and the other one blue is ℓ -regular and bipartite, thus covering the ℓ parallel edges of $W(k, m, \ell, p, q)$ (Proposition 8).
2. In case 2, the input graph G covers $W(1, 0, 1, 0, 1)$ only if G is 2-regular. If this holds, then G is a disjoint union of cycles, and it is easy to see that a cycle covers $W(1, 0, 1, 0, 1)$ if and only if its length is divisible by 4. For the subcase of $k = q = 0$, see the next point.
3. The input graph G covers $W(0, 0, \ell, 0, 0)$ only if it is a bipartite ℓ -regular graph without semi-edges, but in that case it does cover $W(0, 0, \ell, 0, 0)$, as follows from Proposition 8.

4.2 NP-hardness for non-regular target graphs

► **Proposition * 15.** *Let the parameters k, m, p, q be such that $k + 2m \neq 2p + q$, and ($(k \geq 3$ or $k = 2$ and $m \geq 1)$, or $(q \geq 3$ or $q = 2$ and $p \geq 1)$). Then the $W(k, m, \ell, p, q)$ -COVER problem is NP-complete.*

Proof sketch. The proof essentially relies on the reductions from the preceding section. The parameters ensure that after deleting the ℓ ordinary edges from the target graph, we end up with two graphs, $F(k, m)$ and $F(p, q)$, where at least one of them identifies one of the hard cases of covering one-vertex graphs. We then utilize a special gadget by which we connect the vertices of instances of $F(k, m)$ -COVER and $F(p, q)$ -COVER to get a graph G' . We further claim that we can decide both of these instances if and only if G' covers $W(k, m, \ell, p, q)$. The argument is significantly simplified by the determination of images of vertices due to the different degrees of vertices in the target graph. ◀

4.3 NP-hardness for connected regular target graphs

The aim of this subsection is to conclude the proof of Theorem 14 by showing the NP-hardness for the case of $\ell \geq 1$ and $k + 2m = 2p + q$. We will actually prove a result which is more general in two directions. Firstly, we formulate the result in the language of colorings of vertices, and secondly, we prove the hardness for bipartite inputs. This might seem surprising, as we have seen in Section 2 that bipartite graphs can make things easier. Moreover, this strengthening in fact allows us to prove the result in a unified, and hence simpler, way.

Note that the following definition of a relaxation of usual proper 2-coloring resembles the so-called *defective 2-coloring* (see survey of Wood [50]). However, the definitions are not equivalent.

► **Definition 16.** A (b, c) -coloring of a graph is a 2-coloring of its vertices such that every vertex has b neighbors of its own color and c neighbors of the other color.

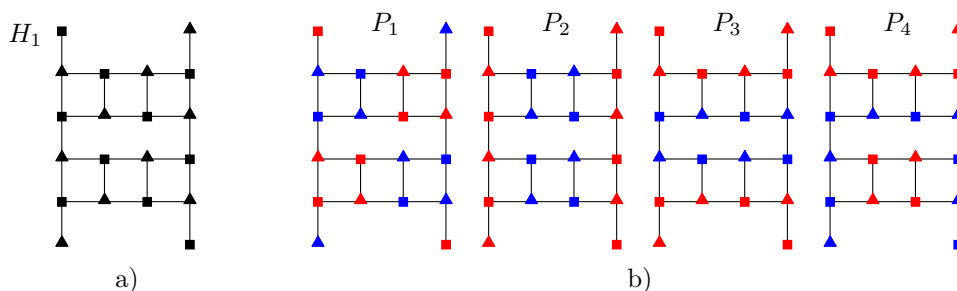
► **Observation 17.** For any parameters k, m, ℓ, p, q such that $k + 2m = 2p + q$, a bipartite graph G with no semi-edges covers $W(k, m, \ell, p, q)$ if and only if it allows a $(k + 2m, \ell)$ -coloring.

Proof. On one hand, any graph covering projection from G to $W(k, m, \ell, p, q)$ induces a $(k + 2m, \ell)$ -coloring of G , provided $k + 2m = 2p + q$. On the other hand, a $(k + 2m, \ell)$ -coloring of G is a degree-obedient vertex mapping from G to $W(k, m, \ell, p, q)$, again provided that $k + 2m = 2p + q$. If G is bipartite and has no semi-edges, then this mapping can be extended to a graph covering projection by Theorem 10. ◀

In view of the previous observation, we will be proving the NP-hardness results for the problem (b, c) -COLORING which takes a graph G on input and asks if G allows a (b, c) -coloring.

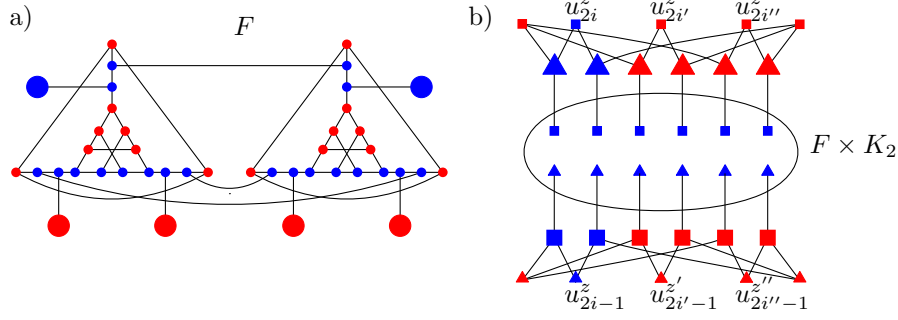
► **Theorem* 18.** For every pair of positive integers b, c such that $b + c \geq 3$, the (b, c) -COLORING problem is NP-complete even for simple bipartite graphs.

Proof sketch. First observe that the (b, c) -COLORING and (c, b) -COLORING problems are polynomially equivalent on bipartite graphs, as the colorings are mutually interchangeable by switching the colors in one class of the bi-partition. Thus we may consider only $b \geq c$.

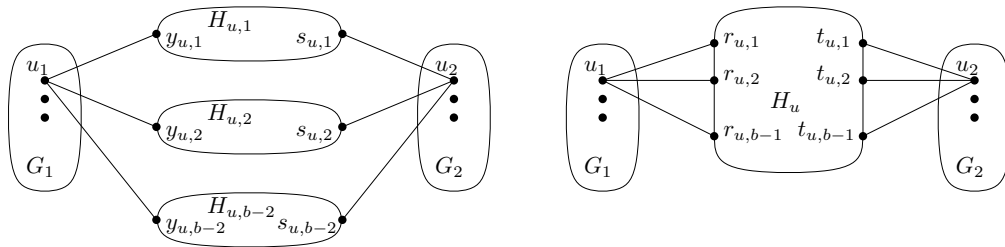


■ **Figure 5** A 20-vertex auxiliary graph H_1 , used in the first part of the proof of Theorem 18, and its possible partial $(2, 1)$ -colorings.

NP-hardness of the $(2, 1)$ -COLORING is proved by a reduction from NAE-3-SAT [25] by using three kinds of building blocks: a clause gadget (here $K_{1,3}$), a vertex gadget enforcing the same color on selected subset of vertices, and a garbage collection that allows to complete the coloring to a cubic graph, that as a part contains the vertex and clause gadgets linked together to represent a given instance of NAE-3-SAT. This reduction is the actual core of the proof, and is briefly sketched in Figures 5 and 6. The former one shows a special gadget



■ **Figure 6** Garbage collection and the overall construction for the first part of Theorem 18. Clause gadgets are in the corners of the figure b).



■ **Figure 7** An illustration of the constructions used in the proof of Theorem 18; a reduction to $(b, 1)$ -COLORING on the left, and a reduction to (b, c) -COLORING on the right.

H_1 used in the color-enforcing constructions of this reduction. For every variable, copies of H_1 are concatenated into a chain, whose one side is connected to ensure that the coloring P_2 (or its inverse) is the only admissible $(2,1)$ -coloring, the other side transfers this information as the truth valuation of the variable to the clause gadgets of clauses containing it. The latter figure sketches the garbage collection and the overall construction of the reduction.

The result on $(2,1)$ -COLORING, in particular, implies that $W(0, 2, 1, 2, 0)$ -COVER is NP-complete for simple bipartite input graphs, whereas the semi-edgeless dumbbell graph $W(0, 2, 1, 2, 0)$ is the smallest semi-edgeless graph whose covering is NP-complete.

Further on, we reduce $(2, 1)$ -COLORING to $(b, 1)$ -COLORING by using two copies of the instance of $(2, 1)$ -COLORING and linking them together by suitable graphs called bridges, that enforce replication of colors for the desired coloring. See a brief sketch in Fig. 7 (left). In view of the initial observation, at this point we know that $(b, 1)$ -COLORING and $(1, b)$ -COLORING are NP-complete on bipartite inputs for all $b \geq 2$.

Then we reduce $(1, c)$ -COLORING to (b, c) -COLORING with $b > c$. Again we take two copies of an instance of $(1, c)$ -COLORING, say a $(1 + c)$ -regular graph G . As sketched in Fig. 7 (right), we construct an auxiliary graph H with two vertices of degree $b - 1$ (called the “connector” vertices), all other vertices being of degree $b + c$ (these are called the “inner” vertices). This bridge graph is such that in every two-coloring of its vertices, such that all inner vertices have exactly b neighbors of their own color and exactly c neighbors of the opposite color, while the connector vertices have at most b neighbors of their own color and at most c neighbors of the opposite color, in every such a coloring the connector vertices and their neighbors always get the same color. And, moreover, such a coloring exists. We then take two copies of G and for every vertex of G , identify its copies with the connector vertices of a copy of the bridge graph (thus we have as many copies of the bridge graph as is the number of vertices of G). The above stated properties of the bridge graph guarantee that the new graph allows a (b, c) -coloring if and only if G allows a $(1, c)$ -coloring.

It is worth mentioning that we have provided two different constructions of the bridge gadget. A general one for the case of $b \geq c + 2$ and a specific one for the case of $b = c + 1$. It is a bit surprising that the case analysis needed to prove the properties of the bridge graph is much more involved for the specific construction in the case of $b = c + 1$.

Finally, for (b, b) -COLORING with $b \geq 2$ we establish a completely different reduction from a special variant of satisfiability $(k\text{-in-}2k)$ -SAT $_q$, a generalization of NAE-3-SAT. ◀

Theorem 18 and Observation 17 imply the following proposition, which concludes the proof of Theorem 14.

► **Proposition 19.** *The $W(k, m, \ell, p, q)$ -COVER problem is NP-complete for simple bipartite input graphs for all parameter sets such that $k + 2m = 2p + q \geq 1$, $\ell \geq 1$, and $k + 2m + \ell \geq 3$.*

5 Conclusion

The main goal of this paper is to initiate the study of the computational complexity of covering graphs with semi-edges. We have exhibited a new level of difficulty that semi-edges bring to coverings by showing a connection to edge-colorings. We have presented a complete classification of the computational complexity of covering graphs with at most two vertices, which is already a quite nontrivial task. In the case of one-vertex target graphs, the problem becomes polynomial-time solvable if the input graph is bipartite, while in the case of two-vertex target graphs, bipartiteness of the input graphs does not help. This provides a strengthening of known results of covering two-vertex graphs without semi-edges.

It is worth noting that the classification in [30] concerns a more general class of *colored mixed* (multi)graphs. I.e., graphs which may have both directed and undirected edges and whose edges come with assigned colors which must be preserved by the covering projections. It turns out that covering a two-vertex (multi)graph is NP-hard if and only if it is NP-hard for at least one of its maximal monochromatic subgraphs. It can be shown that the same holds true when semi-edges are allowed (note that all semi-edges must be undirected only).

We end up with an intriguing open problem.

► **Problem.** *Do there exist graphs H_1 and H_2 , both without semi-edges, such that H_1 covers H_2 , and such that the H_1 -COVER is polynomial-time solvable and H_2 -COVER is NP-complete?*

If semi-edges are allowed, then $H_1 = W(0, 0, 3, 0, 0)$ and $H_2 = F(3, 0)$ is such a pair. All further examples that we can obtain generalize this observation. They are unique in the sense that NP-completeness of H_2 -COVER follows from the NP-completeness of the edge-colorability problem of general graphs which becomes polynomially solvable for bipartite instances.

References

- 1 James Abello, Michael R. Fellows, and John C. Stillwell. On the complexity and combinatorics of covering finite complexes. *Australian Journal of Combinatorics*, 4:103–112, 1991.
- 2 Dana Angluin. Local and global properties in networks of processors. *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 82–93, 1980.
- 3 Dana Angluin and A. Gardiner. Finite common coverings of pairs of regular graphs. *Journal of Combinatorial Theory B*, 30:184–187, 1981.
- 4 Dan Archdeacon. Two graphs without planar covers. *Journal of Graph Theory*, 41(4):318–326, 2002.
- 5 Norman Biggs. *Algebraic Graph Theory*. Cambridge University Press, 1974.

- 6 Ondřej Bílka, Jozef Jirásek, Pavel Klavík, Martin Tancer, and Jan Volec. On the complexity of planar covering of small graphs. In Petr Kolman and Jan Kratochvíl, editors, *Graph-Theoretic Concepts in Computer Science*, volume 6986 of *Lecture Notes in Computer Science*, pages 83–94. Springer, 2011.
- 7 Hans L. Bodlaender. The classification of coverings of processor networks. *Journal of Parallel Distributed Computing*, 6:166–182, 1989.
- 8 Jérémie Chalopin, Yves Métivier, and Wiesław Zielonka. Local computations in graphs: the case of cellular edge local computations. *Fundamenta Informaticae*, 74(1):85–114, 2006.
- 9 Jérémie Chalopin and Daniël Paulusma. Graph labelings derived from models in distributed computing: A complete complexity classification. *Networks*, 58(3):207–231, 2011.
- 10 Jérémie Chalopin and Daniël Paulusma. Packing bipartite graphs with covers of complete bipartite graphs. *Discrete Applied Mathematics*, 168:40–50, 2014.
- 11 Steven Chaplick, Jiří Fiala, Pim van ’t Hof, Daniël Paulusma, and Marek Tesař. Locally constrained homomorphisms on graphs of bounded treewidth and bounded degree. *Theoretical Computer Science*, 590:86–95, 2015.
- 12 Derek G. Corneil. *Graph Isomorphism*. PhD thesis, University of Toronto, 1968.
- 13 Derek G. Corneil and Calvin C. Gotlieb. An efficient algorithm for graph isomorphism. *Journal of the Association for Computing Machinery*, 17:51–64, 1970.
- 14 Bruno Courcelle and Yves Métivier. Coverings and minors: Applications to local computations in graphs. *European Journal of Combinatorics*, 15:127–138, 1994.
- 15 Dragomir Ž. Djoković. Automorphisms of graphs and coverings. *Journal of Combinatorial Theory B*, 16:243–247, 1974.
- 16 Jiří Fiala and Jan Kratochvíl. Locally injective graph homomorphism: Lists guarantee dichotomy. In Fedor V. Fomin, editor, *WG*, volume 4271 of *Lecture Notes in Computer Science*, pages 15–26. Springer, 2006.
- 17 Jiří Fiala. *Locally injective homomorphisms*. PhD thesis, Charles University, Prague, 2000.
- 18 Jiří Fiala, Pinar Heggenes, Petter Kristiansen, and Jan Arne Telle. Generalized H -coloring and H -covering of trees. *Nordic Journal of Computing*, 10(3):206–224, 2003.
- 19 Jiří Fiala, Pavel Klavík, Jan Kratochvíl, and Roman Nedela. Algorithmic aspects of regular graph covers with applications to planar graphs. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP (1)*, volume 8572 of *Lecture Notes in Computer Science*, pages 489–501. Springer, 2014.
- 20 Jiří Fiala, Pavel Klavík, Jan Kratochvíl, and Roman Nedela. 3-connected reduction for regular graph covers. *European Journal of Combinatorics*, 73:170–210, 2018.
- 21 Jiří Fiala and Jan Kratochvíl. Locally constrained graph homomorphisms — structure, complexity, and applications. *Computer Science Review*, 2(2):97–111, 2008.
- 22 Jiří Fiala and Daniël Paulusma. A complete complexity classification of the role assignment problem. *Theoretical Computer Science*, 1(349):67–81, 2005.
- 23 Jiří Fiala, Pavel Klavík, Jan Kratochvíl, and Roman Nedela. Algorithmic aspects of regular graph covers, 2016. [arXiv:1609.03013](https://arxiv.org/abs/1609.03013).
- 24 Anthony Gardiner. Antipodal covering graphs. *Journal of Combinatorial Theory B*, 16:255–273, 1974.
- 25 Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., New York, 1979.
- 26 Ezra Getzler and Mikhail M Kapranov. Modular operads. *Compositio Mathematica*, 110(1):65–125, 1998.
- 27 Jonathan L. Gross and Thomas W. Tucker. Generating all graph coverings by permutation voltage assignments. *Discrete Mathematics*, 18:273–283, 1977.
- 28 Petr Hliněný. $K_{4,4} - e$ has no finite planar cover. *Journal of Graph Theory*, 21(1):51–60, 1998.
- 29 Petr Hliněný and Robin Thomas. On possible counterexamples to Negami’s planar cover conjecture. *Journal of Graph Theory*, 46(3):183–206, 2004.

- 30 Jan Kratochvíl, Andrzej Proskurowski, and Jan Arne Telle. Covering directed multigraphs I. colored directed multigraphs. In Rolf H. Möhring, editor, *WG*, volume 1335 of *Lecture Notes in Computer Science*, pages 242–257. Springer, 1997.
- 31 Jan Kratochvíl, Andrzej Proskurowski, and Jan Arne Telle. Covering directed multigraphs II. when 2-SAT helps. Technical Report 1997–354, KAM-DIMATIA Preprint Series, 1997.
- 32 Jan Kratochvíl, Andrzej Proskurowski, and Jan Arne Telle. Covering regular graphs. *Journal of Combinatorial Theory, Series B*, 71(1):1–16, 1997.
- 33 Jan Kratochvíl, Andrzej Proskurowski, and Jan Arne Telle. Complexity of graph covering problems. *Nordic Journal of Computing*, 5:173–195, 1998.
- 34 Jan Kratochvíl, Jan Arne Telle, and Marek Tesař. Computational complexity of covering three-vertex multigraphs. *Theoretical Computer Science*, 609:104–117, 2016.
- 35 Petter Kristiansen and Jan Arne Telle. Generalized H -coloring of graphs. In D. T. Lee and Shang-Hua Teng, editors, *ISAAC*, volume 1969 of *Lecture Notes in Computer Science*, pages 456–466. Springer, 2000.
- 36 Jin Ho Kwak and Roman Nedela. Graphs and their coverings. *Lecture Notes Series*, 17, 2007.
- 37 Frank Thomas Leighton. Finite common coverings of graphs. *Journal of Combinatorial Theory B*, 33:231–238, 1982.
- 38 Daniel Leven and Zvi Galil. NP completeness of finding the chromatic index of regular graphs. *Journal of Algorithms*, 4:35–44, 1983.
- 39 Igor Litovsky, Yves Métivier, and Wiesław Zielonka. The power and the limitations of local computations on graphs. In Ernst W. Mayr, editor, *WG*, volume 657 of *Lecture Notes in Computer Science*, pages 333–345. Springer, 1992.
- 40 Laszló Lovász and Michael. D. Plummer. *Matching Theory*. Akadémiai Kiadó, Budapest, 1986.
- 41 Gary MacGillivray and Jacobus Swarts. The complexity of locally injective homomorphisms. *Discrete Mathematics*, 310(20):2685–2696, 2010. Graph Theory — Dedicated to Carsten Thomassen on his 60th Birthday.
- 42 Aleksander Malnič, Dragan Marušič, and Primož Potočnik. Elementary abelian covers of graphs. *Journal of Algebraic Combinatorics*, 20(1):71–97, 2004.
- 43 Aleksander Malnič, Roman Nedela, and Martin Škoviera. Lifting graph automorphisms by voltage assignments. *European Journal of Combinatorics*, 21(7):927–947, 2000.
- 44 Alexander D. Mednykh and Roman Nedela. *Harmonic Morphisms of Graphs: Part I: Graph Coverings*. Vydavateľstvo Univerzity Mateja Bela v Banskej Bystrici, 1st edition, 2015.
- 45 Roman Nedela and Martin Škoviera. Regular embeddings of canonical double coverings of graphs. *Journal of Combinatorial Theory, Series B*, 67(2):249–277, 1996.
- 46 Seiya Negami. Graphs which have no planar covering. *Bulletin of the Institute of Mathematics, Academia Sinica*, 16(4):377–384, 1988.
- 47 Julius Petersen. Die theorie der regulären graphs. *Acta Mathematica*, 15:193–220, 1891.
- 48 Gerhard Ringel. *Map color theorem*, volume 209. Springer, Berlin, 1974.
- 49 Sam Shepherd, Giles Gardam, and Daniel J. Woodhouse. Two generalisations of Leighton’s theorem, 2019. [arXiv:1908.00830](https://arxiv.org/abs/1908.00830).
- 50 David R. Wood. Defective and clustered graph colouring. *Electronic Journal of Combinatorics*, 2018. [doi:10.37236/7406](https://doi.org/10.37236/7406).
- 51 Daniel J. Woodhouse. Revisiting Leighton’s theorem with the Haar measure, 2018. [arXiv:1806.08196](https://arxiv.org/abs/1806.08196).

Coherent Control and Distinguishability of Quantum Channels via PBS-Diagrams

Cyril Branciard  

Université Grenoble Alpes, CNRS, Grenoble INP, Institut Néel, F-38000 Grenoble, France

Alexandre Clément  

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Mehdi Mhalla  

Université Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France

Simon Perdrix  

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract

Even though coherent control of quantum operations appears to be achievable in practice, it is still not yet well understood. Among theoretical challenges, standard completely positive trace preserving (CPTP) maps are known not to be appropriate to represent coherently controlled quantum channels. We introduce here a graphical language for coherent control of general quantum channels inspired by practical quantum optical setups involving polarising beam splitters (PBS). We consider different situations of coherent control and disambiguate CPTP maps by considering purified channels, an extension of Stinespring’s dilation.

First, we show that in classical control settings, the observational equivalence classes of purified channels correspond to the standard definition of quantum channels (CPTP maps). Then, we propose a refinement of this equivalence class generalising the “half quantum switch” situation, where one is allowed to coherently control which quantum channel is applied; in this case, quantum channel implementations can be distinguished using a so-called transformation matrix. A further refinement characterising observational equivalence with general extended PBS-diagrams as contexts is also obtained. Finally, we propose a refinement that could be used for more general coherent control settings.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Axiomatic semantics; Theory of computation → Categorical semantics; Hardware → Quantum computation; Hardware → Quantum communication and cryptography

Keywords and phrases Quantum Computing, Diagrammatic Language, Quantum Control, Polarising Beam Splitter, Categorical Quantum Mechanics, Quantum Switch

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.22

Related Version *Full Version:* <https://arxiv.org/abs/2103.02073> [3]

Funding This work is funded by ANR-17-CE25-0009 SoftQPro, ANR-17-CE24-0035 VanQuTe, PIA-GDN/Quantex, LUE/UOQ, and by “*Investissements d’avenir*” (ANR-15-IDEX-02) program of the French National Research Agency.

1 Introduction

Unlike the usual sequential and parallel compositions, coherent control allows one to perform two or more quantum evolutions in superposition. It is fairly easy with quantum optics – an important player in the development of quantum technologies – to construct setups that perform some coherent control. A polarising beam splitter (PBS) precisely allows one to do that: by reflecting for instance horizontally polarised particles and transmitting vertically polarised ones, it lets the polarisation control the path, and thereby the physical devices



© Cyril Branciard, Alexandre Clément, Mehdi Mhalla, and Simon Perdrix;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 22; pp. 22:1–22:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

encountered, in a coherent way [10, 16]. This finds some interesting applications for quantum information processing (e.g., for error filtration [11]), including the ability to perform some operations in an indefinite causal order, as for instance in the so-called quantum switch [6]. Intuitively, given two quantum operations A and B and a control qubit, a quantum switch consists in applying A followed by B (resp. B followed by A) when the control qubit is in state $|0\rangle$ (resp. $|1\rangle$). When the control qubit is in a superposition, we get a superposition of the two possible orders. Quantum switch can be used to speed up information processing tasks, e.g. deciding whether two operators are commuting or anticommuting [5, 2]. Actual implementations of the quantum switch have been experimentally realised [12].

General quantum evolutions – a.k.a. quantum channels – are commonly represented as completely positive trace preserving (CPTP) maps. CPTP maps can naturally be composed in sequence and in parallel. However, it has been realised that the description of quantum channels in terms of CPTP maps is not appropriate for some particular setups involving coherent control [15, 1, 7, 13]. One indeed needs some more information about their practical implementation to unambiguously determine the behaviour of such setups, and it was recently proposed to complete the description of channels by so-called transformation matrices [1], or vacuum extensions [7, 13].

Here we consider a general class of setups involving PBS, and study how these can be used to coherently control quantum channels. We build upon the graphical language of PBS-diagrams introduced in [8], in which the controlled operations were “pure” (typically, unitary), and extend it to allow for the control of more general quantum channels. As the description of channels as CPTP maps is inadequate here, we propose to work with *purified channels* based on a unitary extension of Stinespring’s dilation [17].

We address the question of the observational equivalence of purified channels, and show that different purified channels can be indistinguishable. To do so, we use PBS-diagrams to formalise three kinds of *contexts*: when the context is PBS-free, we recover that two purified channels are indistinguishable if and only if they lead to the same CPTP map. When the context allows for PBS but no polarisation flips, we recover the characterisation in terms of superoperators and transformation matrices which was introduced for a particular setup [1]. When we allow for arbitrary contexts, we obtain a characterisation of observational equivalence involving “second-level” superoperators and transformation matrices. We finally open the discussion to more general coherent-control settings, and propose a refined equivalence relation as a candidate for characterising channel (in)distinguishability in such scenarios.

The omitted proofs are available in the full version of the paper [3].

2 PBS-diagrams

PBS-diagrams were introduced in [8] as a language for coherent control of “pure” quantum evolutions. They aim at describing practical scenarios where a flying particle goes through an experimental setup, and is routed via polarising beam splitters. In addition to its polarisation, the particle carries some “data” register, whose state is described in some Hilbert space \mathcal{H} , and on which a number “pure” linear (typically, unitary) operators are applied.

Here we shall enrich the pure PBS-diagram language so as to incorporate the coherent control of more general quantum channels. To this purpose, we start by defining an abstract version of PBS-diagrams that we call *bare diagrams*, and which we equip with a word path semantics describing the trajectory and change of polarisation of a particle that enters the diagram through some given input wire: the word path semantics gives its new polarisation and position at the output of the diagram, together with a word over some alphabet describing

the sequence of *bare gates* – where the quantum channels we want to control are located – crossed. Subscribing to the idea that any general quantum operation can be seen as a unitary evolution of the system under consideration and its environment, we then define *purified channels*, which can be coherently controlled in a similar way to the PBS-diagrams of [8]. Replacing bare gates with purified channels, we obtain an extension¹ of the graphical language of [8], which we call *extended PBS-diagrams* and which we equip with a quantum semantics obtained after discarding the (inaccessible) environments of all gates.

2.1 Bare PBS-diagrams

2.1.1 Syntax

A *bare PBS-diagram* is made of polarising beam splitters \bowtie , polarisation flips \ominus , and bare gates \square_a . Every bare gate is indexed by a unique label (here, a) used to identify the gate in the diagram. These building blocks are connected via wires represented using the identity — or the swap \bowtie . The empty diagram is denoted by $[\]$. Diagrams can be combined by means of sequential composition \circ , parallel composition \oplus ,² and trace $Tr(\cdot)$, which represents a feedback loop.

We define a typing judgement $\Gamma \vdash D : n$, where Γ is the alphabet containing all gate indices,³ to guarantee that the diagrams are well-formed – in particular, that the gate indices are unique – using a linear typing discipline:

► **Definition 1** (Bare PBS-diagram). *A bare PBS-diagram $\Gamma \vdash D : n$ (with $n \in \mathbb{N}$) is inductively defined as:*

$$\emptyset \vdash [\] : 0 \quad \emptyset \vdash \text{—} : 1 \quad \emptyset \vdash \ominus : 1 \quad \emptyset \vdash \bowtie : 2 \quad \emptyset \vdash \bowtie : 2 \quad \{a\} \vdash \square_a : 1$$

$$\frac{\Gamma_1 \vdash D_1 : n \quad \Gamma_2 \vdash D_2 : n \quad \Gamma_1 \cap \Gamma_2 = \emptyset}{\Gamma_1 \cup \Gamma_2 \vdash D_2 \circ D_1 : n} \quad \frac{\Gamma_1 \vdash D_1 : n_1 \quad \Gamma_2 \vdash D_2 : n_2 \quad \Gamma_1 \cap \Gamma_2 = \emptyset}{\Gamma_1 \cup \Gamma_2 \vdash D_1 \oplus D_2 : n_1 + n_2} \quad \frac{\Gamma \vdash D : n + 1}{\Gamma \vdash Tr(D) : n}$$

Graphical representation. PBS-diagrams form a graphical language: compositions and trace are respectively depicted as follows (for diagrams generically depicted as \boxed{D}):

$$\boxed{D_2} \circ \boxed{D_1} = \boxed{D_1} \boxed{D_2} \quad \boxed{D_1} \oplus \boxed{D_2} = \begin{array}{c} \boxed{D_1} \\ \boxed{D_2} \end{array} \quad Tr(\boxed{D}) = \begin{array}{c} \boxed{D} \\ \boxed{D} \end{array}$$

Examples of bare PBS-diagrams are given in Fig. 1 below. Note that two *a priori* distinct constructions, like for instance $Tr(\square_a \oplus \bowtie)$ and $\square_a \oplus Tr(\bowtie)$, can lead to the same graphical representation \square_a . To avoid ambiguity, we define diagrams modulo a structural

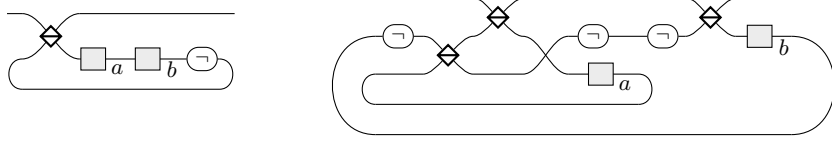
congruence detailed in Appendix A. Roughly speaking, the structural congruence guarantees that (i) two constructions leading to the same graphical representation are equivalent, and (ii) a diagram can be deformed at will (without changing its topology), e.g.:

$$\begin{array}{c} \bowtie \\ \text{—} \end{array} = \text{—} \quad \begin{array}{c} \boxed{D_1} \\ \boxed{D_2} \end{array} \bowtie = \begin{array}{c} \boxed{D_2} \\ \boxed{D_1} \end{array} \bowtie \quad \begin{array}{c} \text{—} \\ \text{—} \end{array} = \text{—} \quad \begin{array}{c} \boxed{D_1} \\ \boxed{D_2} \end{array} = \begin{array}{c} \boxed{D_2} \\ \boxed{D_1} \end{array}$$


¹ Strictly speaking, the PBS-diagrams of [8] did not require the operations inside the gates to be unitary, while here we impose such a restriction *a priori*. One could however also consider non-unitary operations in our framework here, although one would lose our motivation based on the unitary extension of Stinespring's dilation.

² Denoted \otimes in [8]. Here we change the notation to reflect how the parallel composition affects the structure of the Hilbert space describing the position of the particle (see Section 2.2).

³ We may write simply $D : n$, or even just D , when Γ is not relevant or is clear from the context.



■ **Figure 1** Two examples of bare PBS-diagrams, with the same word path semantics: $(D, \uparrow, 0) \xrightarrow{abab} (\uparrow, 0)$ and $(D, \rightarrow, 0) \xrightarrow{\epsilon} (\rightarrow, 0)$.

Note in particular that the length of the wires does not matter. Physically, if these diagrams were to be realised in practical setups, this would mean that the experiment should be insensible to the time at which the particle would go through the various elements; if needed one could always add (possibly polarisation-dependent) delay lines (e.g., ) to correct for a possible time mismatch between different paths.

2.1.2 Word path semantics

The word path semantics describes the trajectory of a particle which enters a bare PBS-diagram $\Gamma \vdash D : n$ with a polarisation in the standard basis state $c \in \{\rightarrow, \uparrow\}$ (horizontal or vertical) and from a definite position $p \in [n] := \{0, \dots, n-1\}$. Because of the polarising beam splitters, the trajectory of the particle depends on its polarisation: we take it to be reflected when the polarisation is horizontal, and transmitted when the polarisation is vertical. The “negation” \ominus flips the polarisation, while the gates do not act on the polarisation. The word path semantics of a diagram describes, given an initial polarisation and position, the final polarisation and position together with the sequence of gates – represented by a word over Γ – that the particle goes through:

► **Definition 2** (Word path semantics). *Given a bare PBS-diagram $\Gamma \vdash D : n$, a polarisation $c \in \{\rightarrow, \uparrow\}$ and a position $p \in [n]$, let $(D, c, p) \xrightarrow{w} (c', p')$ with $w \in \Gamma^*$ a word over Γ (or just $(D, c, p) \Rightarrow (c', p')$ for the empty word $w = \epsilon$) be inductively defined as follows:*

$$(\rightarrow, c, 0) \Rightarrow (c, 0) \quad (\ominus \rightarrow, \uparrow, 0) \Rightarrow (\rightarrow, 0) \quad (\ominus \rightarrow, \rightarrow, 0) \Rightarrow (\uparrow, 0)$$

$$(\rightarrow \curvearrowright, c, p) \Rightarrow (c, 1-p) \quad (\rightarrow \curvearrowleft, \rightarrow, p) \Rightarrow (\rightarrow, p) \quad (\rightarrow \curvearrowleft, \uparrow, p) \Rightarrow (\uparrow, 1-p)$$

$$(\rightarrow \square_a, c, 0) \xrightarrow{a} (c, 0) \quad \frac{(D_1, c, p) \xrightarrow{w_1} (c', p') \quad (D_2, c', p') \xrightarrow{w_2} (c'', p'')}{(D_2 \circ D_1, c, p) \xrightarrow{w_1 w_2} (c'', p'')} (\circ)$$

$$\frac{D_1 : n_1 \quad p < n_1 \quad (D_1, c, p) \xrightarrow{w} (c', p')}{(D_1 \oplus D_2, c, p) \xrightarrow{w} (c', p')} (\oplus_1) \quad \frac{D_1 : n_1 \quad p \geq n_1 \quad (D_2, c, p-n_1) \xrightarrow{w} (c', p')}{(D_1 \oplus D_2, c, p) \xrightarrow{w} (c', p'+n_1)} (\oplus_2)$$

$$\frac{D : n+1 \quad \forall i \in \{0, \dots, k\}, (D, c_i, p_i) \xrightarrow{w_i} (c_{i+1}, p_{i+1}) \quad (p_{i+1} = n) \Leftrightarrow (i < k)}{(Tr(D), c_0, p_0) \xrightarrow{w_0 \dots w_k} (c_{k+1}, p_{k+1})} (\mathbb{T}_k)$$

with $k = 0, 1$, and 2.

We denote by $w_{c,p}^D \in \Gamma^*$ the word, $c_{c,p}^D \in \{\uparrow, \rightarrow\}$ the polarisation, and $p_{c,p}^D \in [n]$ the position s.t. $(D, c, p) \xrightarrow{w_{c,p}^D} (c_{c,p}^D, p_{c,p}^D)$.

The word path semantics is invariant modulo structural congruence (i.e., diagram deformation). Moreover, note that despite the traces which form feedback loops, the word path semantics is well-defined.⁴ Indeed, a particle entering the diagram through some input wire cannot go through a feedback loop (or any other part of the diagram) twice with the same polarisation, which justifies that k only needs to go up to 2 in Rule (\mathbb{T}_k) above. Intuitively, if a particle goes twice in a feedback loop with the same polarisation then it will loop forever; but because of time symmetry this also means that the particle went through the feedback loop infinitely many times in the past, which contradicts the fact that it entered through an input wire. See Appendix B for details about the formal proofs of these facts.

For similar reasons, each gate cannot appear more than twice along any path, or even in the family of all the possible paths of a diagram:

► **Proposition 3.** *Given a bare PBS-diagram $\Gamma \vdash D : n$, $\forall a \in \Gamma$, one has $\sum_{c \in \{\rightarrow, \uparrow\}, p \in [n]} |w_{c,p}^D|_a \leq 2$, where $|w|_a$ denotes the number of occurrences of a in the word w . Moreover, if D is \ominus -free then for any c one has $\sum_{p \in [n]} |w_{c,p}^D|_a \leq 1$.*

The converse is also true:

► **Proposition 4.** *For any family of words $\{w_{c,p}\}_{(c,p) \in \{\rightarrow, \uparrow\} \times [n]}$ such that every letter appears at most twice in the whole family, there exists a bare PBS-diagram $D : n$ such that $w_{c,p} = w_{c,p}^D$ for all c, p . Furthermore if for any $c \in \{\rightarrow, \uparrow\}$, every letter appears at most once in $\{w_{c,p}\}_{p \in [n]}$, the bare PBS-diagram D can be chosen \ominus -free.*

Note that the proof of Proposition 4 is constructive. For instance, the family $\{w_{\uparrow,0} = abab, w_{\rightarrow,0} = \epsilon\}$ can be obtained from the diagram of Fig. 1 (Right). The solution is not unique in general and there is actually a simpler diagram, see Fig. 1 (Left), with the same word path semantics.

2.2 Extended PBS-diagrams

We will now introduce extended PBS-diagrams by filling every bare gate with the description of a quantum channel. As recalled in the introduction, however, defining the coherent control of general channels (as we wish to do with PBS-diagrams) in an unambiguous way is not trivial. Here we propose to do so through the notion of purified channels, which are an extension of Stinespring’s dilation of quantum channels [17].

2.2.1 Purified channels

A standard paradigm for quantum channels acting on a Hilbert space \mathcal{H} is to describe them as CPTP maps, or superoperators $\mathcal{L}(\mathcal{H}) \rightarrow \mathcal{L}(\mathcal{H})$,⁵ where $\mathcal{L}(\mathcal{H})$ denotes the set of linear operators on \mathcal{H} . As exemplified e.g. in [15, 1], this representation is however ambiguous when it comes to describing quantum coherent control: two quantum channels with the same superoperator can behave differently in a coherent-control setting.

A possible way to overcome this issue is to “go to the Church of the larger Hilbert space”, according to which any quantum channel can be interpreted as a pure quantum operation acting on both the quantum system and an environment. Mathematically, this corresponds to Stinespring’s dilation theorem [17], which states that any CPTP map acting

⁴ Definition 2 does not provide any word path semantics for diagrams of type $D : 0$. In fact, no word path semantics needs to be defined for such diagrams, as there is no position p defining any input wire. Note also that for diagrams $D : n$ containing fully closed subdiagrams (e.g., of the form $D = D_1 \oplus D_2$ with $D_2 : 0$), the semantics does not depend on these fully closed subdiagrams.

⁵ As this is the case of interest in PBS-diagrams (with \mathcal{H} corresponding to the data register), we consider here channels with the same input and output Hilbert spaces.

on a Hilbert space \mathcal{H} can be implemented with an isometry $V : \mathcal{H} \rightarrow \mathcal{H} \otimes \mathcal{E}$, where \mathcal{E} denotes the Hilbert space attached to the environment, followed by a partial trace of the latter. Note that in this representation, the isometry V can be understood as encoding both the creation of the environment \mathcal{E} and the evolution of the joint system $\mathcal{H} \otimes \mathcal{E}$. Indeed, V can always be decomposed into an environment initialisation $|\varepsilon\rangle \in \mathcal{E}$ and a unitary evolution $U : \mathcal{H} \otimes \mathcal{E} \rightarrow \mathcal{H} \otimes \mathcal{E}$ such that $V = U(I_{\mathcal{H}} \otimes |\varepsilon\rangle)$, where $I_{\mathcal{H}}$ denotes the identity operator over \mathcal{H} . In our approach to defining coherent control for quantum channels, we will precisely abide by this description in terms of unitary purifications, which we formalise as follows:

► **Definition 5 (Purified channel).** *Given a Hilbert space \mathcal{H} , a purified \mathcal{H} -channel (or simply purified channel, for short) is a triplet $[U, |\varepsilon\rangle, \mathcal{E}]$, where \mathcal{E} is the local environment Hilbert space, $|\varepsilon\rangle \in \mathcal{E}$ is the environment initial state, and $U : \mathcal{H} \otimes \mathcal{E} \rightarrow \mathcal{H} \otimes \mathcal{E}$ is a unitary operator representing the evolution of the joint system. We denote the set of purified \mathcal{H} -channels by $\mathfrak{C}(\mathcal{H})$.*

As seen above, it directly follows from Stinespring’s dilation theorem that any CPTP map $\mathcal{L}(\mathcal{H}) \rightarrow \mathcal{L}(\mathcal{H})$ can be represented by a purified \mathcal{H} -channel, which is however not unique. Reciprocally, with any purified \mathcal{H} -channel $[U, |\varepsilon\rangle, \mathcal{E}]$, we naturally associate the CPTP map $\mathcal{S}_{[U, |\varepsilon\rangle, \mathcal{E}]}^{(1)} : \mathcal{L}(\mathcal{H}) \rightarrow \mathcal{L}(\mathcal{H}) = \rho \mapsto \text{Tr}_{\mathcal{E}}(U(\rho \otimes |\varepsilon\rangle\langle\varepsilon|)U^\dagger)$, where $\text{Tr}_{\mathcal{E}}$ denotes the partial trace over \mathcal{E} , and which we shall represent graphically, using the circuit notations of Appendix C,⁶ as follows: $\mathcal{S}_{[U, |\varepsilon\rangle, \mathcal{E}]}^{(1)} = |\varepsilon\rangle \text{---} \boxed{U} \text{---} |\varepsilon\rangle$.

One may however not trace out the environment straight away. In fact, decomposing Stinespring’s dilation into an environment state initialisation and a unitary evolution of the joint system, as we did above, allows one to apply the same channel several times in a coherent manner if a particle goes through a gate several times. In that case we will consider that the same unitary is applied each time, without re-initialising the environment state (which we assume to not evolve between two applications of the channel).

2.2.2 From bare to extended PBS-diagrams

We are now in a position to define extended PBS-diagrams of type $\mathcal{H}^{(n)}$, which are essentially bare PBS-diagrams of type n , where the gate indices are replaced by purified \mathcal{H} -channels. Hence, instead of bare gates $\text{---} \boxed{a} \text{---}$, an extended PBS-diagram contains gates of the form $\text{---} \boxed{[U, |\varepsilon\rangle]} \text{---}$, parametrised by a purified channel $[U, |\varepsilon\rangle, \mathcal{E}] \in \mathfrak{C}(\mathcal{H})$ (where the Hilbert space \mathcal{E} is not represented explicitly, in order not to overload the diagrams).

This leads to the following inductive definition:

► **Definition 6 (Extended PBS-diagram).** *An extended PBS-diagram $D : \mathcal{H}^{(n)}$ (with $n \in \mathbb{N}$) is inductively defined as:*

$$\begin{array}{c}
 \boxed{} : \mathcal{H}^{(0)} \quad \text{---} : \mathcal{H}^{(1)} \quad \boxed{\ominus} : \mathcal{H}^{(1)} \quad \text{---} \text{---} : \mathcal{H}^{(2)} \quad \boxed{\otimes} : \mathcal{H}^{(2)} \quad \frac{[U, |\varepsilon\rangle, \mathcal{E}] \in \mathfrak{C}(\mathcal{H})}{\boxed{[U, |\varepsilon\rangle]} : \mathcal{H}^{(1)}} \\
 \\
 \frac{D_1 : \mathcal{H}^{(n)} \quad D_2 : \mathcal{H}^{(n)}}{D_2 \circ D_1 : \mathcal{H}^{(n)}} \quad \frac{D_1 : \mathcal{H}^{(n_1)} \quad D_2 : \mathcal{H}^{(n_2)}}{D_1 \oplus D_2 : \mathcal{H}^{(n_1+n_2)}} \quad \frac{D : \mathcal{H}^{(n+1)}}{\text{Tr}(D) : \mathcal{H}^{(n)}}
 \end{array}$$

⁶ To manipulate unitary operations and CPTP maps, it is convenient to use such circuit-like graphical representations, which correspond to standard circuit notations for “pure” operations, supplemented with a ground symbol $\text{---} \lfloor \text{---}$ for the case of CPTP maps; see Appendix C for details.

Extended PBS-diagrams are defined up to the same structural congruence as for bare PBS-diagrams. It is convenient to explicitly define the map which, given a family of purified channels, transforms a bare diagram into the corresponding extended PBS-diagram:⁷

► **Definition 7.** *Given a bare PBS-diagram $\Gamma \vdash D' : n$ and a family of purified \mathcal{H} -channels $\mathcal{G} = ([U_a, |\varepsilon_a\rangle, \mathcal{E}_a])_{a \in \Gamma}$ indexed by elements of Γ , let $[D']_{\mathcal{G}} : \mathcal{H}^{(n)}$ be the extended PBS-diagram inductively defined as $[-\square_a]([U_a, |\varepsilon_a\rangle, \mathcal{E}_a]) = -\overline{[U_a, |\varepsilon_a\rangle]}$, $\forall g \in \{\overline{[\]}, -, \ominus, \bowtie, \overline{\bowtie}\}$, $[g]_{\emptyset} = g$, $[D'_2 \circ D'_1]_{\mathcal{G}_1 \uplus \mathcal{G}_2} = [D'_2]_{\mathcal{G}_2} \circ [D'_1]_{\mathcal{G}_1}$, $[D'_1 \oplus D'_2]_{\mathcal{G}_1 \uplus \mathcal{G}_2} = [D'_1]_{\mathcal{G}_1} \oplus [D'_2]_{\mathcal{G}_2}$ and $[Tr(D')]_{\mathcal{G}} = Tr([D']_{\mathcal{G}})$, where \uplus is the disjoint union.*

For any extended PBS-diagram $D : \mathcal{H}^{(n)}$, there exists a bare diagram $\Gamma \vdash D' : n$ and an indexed family of purified \mathcal{H} -channels \mathcal{G} s.t. $[D']_{\mathcal{G}} = D$. We call D' an *underlying bare diagram* of D (which is unique, up to relabelling of the gates).

2.2.3 Quantum semantics

We now equip the extended PBS-diagrams with a quantum semantics, which is a CPTP map acting on the complete state of the particle that goes through it, i.e., its joint polarisation, position and data state. To describe the quantum semantics of an extended PBS-diagram $D : \mathcal{H}^{(n)}$, it is convenient to rely on an underlying bare diagram $\Gamma \vdash D' : n$ and a family of purified channels \mathcal{G} s.t. $[D']_{\mathcal{G}} = D$ (so as to keep track of the environment spaces and be able to identify them via the bare gate indices).

As we defined them, every purified channel comes with its local environment and a unitary evolution acting on both the data register and its local environment. In order to define the overall evolution of the diagram, we consider the global environment as the tensor product of these local environments, and extend every unitary transformation to a global transformation acting on the data register and the global environment:

► **Definition 8.** *Given an indexed family of purified \mathcal{H} -channels $\mathcal{G} = ([U_a, |\varepsilon_a\rangle, \mathcal{E}_a])_{a \in \Gamma}$, let $\mathcal{E}_{\mathcal{G}} := \bigotimes_{a \in \Gamma} \mathcal{E}_a$, $|\varepsilon_{\mathcal{G}}\rangle := \bigotimes_{a \in \Gamma} |\varepsilon_a\rangle \in \mathcal{E}_{\mathcal{G}}$, and $\forall a \in \Gamma$, let $V_a^{\mathcal{G}} := U_a \bigotimes_{x \in \Gamma \setminus \{a\}} I_{\mathcal{E}_x} \in \mathcal{L}(\mathcal{H} \otimes \mathcal{E}_{\mathcal{G}})$.*

If a particle enters an extended PBS-diagram D with a definite polarisation and position in some basis states $|c\rangle \in \mathbb{C}^{\{\rightarrow, \uparrow\}}$ and $|p\rangle \in \mathbb{C}^{[n]}$, respectively, the sequence of transformations applied to the particle and the global environment when the particle goes through the diagram can be deduced from the word path semantics of the underlying bare diagram D' :

$$|c\rangle \otimes |p\rangle \otimes |\psi\rangle \otimes |\varepsilon_{\mathcal{G}}\rangle \mapsto |c_{c,p}^{D'}\rangle \otimes |p_{c,p}^{D'}\rangle \otimes V_{w_{c,p}^{D'}}^{\mathcal{G}}(|\psi\rangle \otimes |\varepsilon_{\mathcal{G}}\rangle)$$

where $w_{c,p}^{D'}$, $c_{c,p}^{D'}$, and $p_{c,p}^{D'}$ are given by the word path semantics, i.e., $(D', c, p) \xrightarrow{w_{c,p}^{D'}} (c_{c,p}^{D'}, p_{c,p}^{D'})$, and $V_w^{\mathcal{G}}$ is inductively defined as $V_{\epsilon}^{\mathcal{G}} := I_{\mathcal{H} \otimes \mathcal{E}}$ and $\forall a \in \Gamma, \forall w \in \Gamma^*$, $V_{aw}^{\mathcal{G}} := V_w^{\mathcal{G}} V_a^{\mathcal{G}}$.

One can actually consider inputting a particle in an arbitrary initial state (i.e., including superpositions of polarisation and position); the transformation applied by the diagram is then obtained from the one above, by linearity. This leads us to define the following:

► **Definition 9.** *Given a bare PBS-diagram $\Gamma \vdash D' : n$ and a family of purified \mathcal{H} -channels \mathcal{G} indexed with Γ , let*

$$U_{D'}^{\mathcal{G}} := \sum_{c \in \{\rightarrow, \uparrow\}, p \in [n]} |c_{c,p}^{D'}\rangle \langle c| \otimes |p_{c,p}^{D'}\rangle \langle p| \otimes V_{w_{c,p}^{D'}}^{\mathcal{G}}$$

⁷ To clarify which kind of diagram we are dealing with, in this subsection we use primed names (e.g., D') when referring to bare PBS-diagrams, and nonprimed names for extended PBS-diagrams.

The triplet $[U_{D'}^{\mathcal{G}}, |\varepsilon_{\mathcal{G}}\rangle, \mathcal{E}_{\mathcal{G}}]$ is nothing but a purified $(\mathbb{C}^{\{\rightarrow, \uparrow\}} \otimes \mathbb{C}^{[n]} \otimes \mathcal{H})$ -channel, which describes the action of the corresponding extended PBS-diagram on the complete state of the particle. Once the particle exits the diagram, the environments of all purified channels are not accessible anymore. As is well-known, the statistics of any “input/output test”, which consists in preparing an arbitrary input state of the particle and measuring the output in an arbitrary basis, then only depend on the CPTP map (the superoperator) induced by $U_{D'}^{\mathcal{G}}$, above, with all environments initially prepared in the global state $|\varepsilon_{\mathcal{G}}\rangle$, and after tracing out all environment spaces – i.e., using circuit-like notations: $|\varepsilon_{\mathcal{G}}\rangle\langle\varepsilon_{\mathcal{G}}| \text{---} \boxed{U_{D'}^{\mathcal{G}}} \text{---} |_{\parallel}$. This superoperator thus precisely captures input/output (in)distinguishability: two quantum channels have the same superoperator if and only if they are indistinguishable in any input/output test. This provides the ground for our definition of the following quantum semantics:

► **Definition 10** (Quantum Semantics). *Given an extended PBS-diagram $D : \mathcal{H}^{(n)}$, let $\llbracket D \rrbracket : \mathcal{L}(\mathbb{C}^{\{\rightarrow, \uparrow\}} \otimes \mathbb{C}^{[n]} \otimes \mathcal{H}) \rightarrow \mathcal{L}(\mathbb{C}^{\{\rightarrow, \uparrow\}} \otimes \mathbb{C}^{[n]} \otimes \mathcal{H})$ be the superoperator defined as*

$$\llbracket D \rrbracket := \rho \mapsto \text{Tr}_{\mathcal{E}_{\mathcal{G}}}(U_{D'}^{\mathcal{G}}(\rho \otimes |\varepsilon_{\mathcal{G}}\rangle\langle\varepsilon_{\mathcal{G}}|)U_{D'}^{\mathcal{G}\dagger}) = |\varepsilon_{\mathcal{G}}\rangle\langle\varepsilon_{\mathcal{G}}| \text{---} \boxed{U_{D'}^{\mathcal{G}}} \text{---} |_{\parallel}$$

where $\Gamma \vdash D' : n$ is an underlying bare diagram and \mathcal{G} is an indexed family of purified \mathcal{H} -channels s.t. $[D']_{\mathcal{G}} = D$.

Note that the quantum semantics is preserved by the “only topology matters” structural congruence on diagrams. Indeed, it is defined using only the family \mathcal{G} and the word path semantics of its underlying bare diagram D' , which is invariant modulo diagram deformation. It is clear that when deforming D we do not have to change D' and \mathcal{G} , since it suffices to deform D' accordingly.

3 Observational equivalence of purified channels

In this section we address the problem of deciding whether two purified channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$ can be distinguished in an experiment involving coherent control, within the framework of PBS-diagrams just established. We introduce for that the notion of *contexts*, which are extended PBS-diagrams with a “hole”: if for any context, filling its hole with $[U, |\varepsilon\rangle, \mathcal{E}]$ or $[U', |\varepsilon'\rangle, \mathcal{E}']$ leads to diagrams with the same quantum semantics, then the two purified channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$ are indistinguishable within our framework, even with the help of the coherent control provided by extended PBS-diagrams.

3.1 Contexts

A context is an extended PBS-diagram with a *hole*, i.e., a (unique) particular empty gate, without any purified channel specified *a priori*. Equivalently a context can be seen as a bare PBS-diagram partially filled: all but one gate are filled with purified channels. Formally:

- **Definition 11** (Context). *A context $C[\cdot] : \mathcal{H}^{(n)}$ (with $n \in \mathbb{N}$) is inductively defined as follows:*
- The hole gate $\text{---} \square \text{---} : \mathcal{H}^{(1)}$ is a context;
 - If $C[\cdot] : \mathcal{H}^{(n)}$ is a context and $D : \mathcal{H}^{(n)}$ is an extended PBS-diagram then $D \circ C[\cdot] : \mathcal{H}^{(n)}$ and $C[\cdot] \circ D : \mathcal{H}^{(n)}$ are contexts;
 - If $C[\cdot] : \mathcal{H}^{(n)}$ is a context and $D : \mathcal{H}^{(m)}$ is an extended PBS-diagram then $D \oplus C[\cdot] : \mathcal{H}^{(m+n)}$ and $C[\cdot] \oplus D : \mathcal{H}^{(n+m)}$ are contexts;
 - If $C[\cdot] : \mathcal{H}^{(n+1)}$ is a context then $\text{Tr}(C[\cdot]) : \mathcal{H}^{(n)}$ is a context.

Like bare and extended PBS-diagrams, contexts are defined up to structural congruence.

► **Definition 12** (Substitution). *For any context $C[\cdot] : \mathcal{H}^{(n)}$ and any purified \mathcal{H} -channel $[U, |\varepsilon\rangle, \mathcal{E}]$, let $C[U, |\varepsilon\rangle, \mathcal{E}] : \mathcal{H}^{(n)}$ be the extended PBS-diagram obtained by replacing the single hole \square in $C[\cdot]$ by the purified channel $\boxed{U, |\varepsilon\rangle}$.*

After some purified channel is plugged in, contexts allow one to compare the quantum semantics $\llbracket C[U, |\varepsilon\rangle, \mathcal{E}] \rrbracket$ and $\llbracket C[U', |\varepsilon'\rangle, \mathcal{E}'] \rrbracket$ induced by different purified channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$. We consider in the following three subclasses of contexts, depending on the kind of coherent control one may allow to distinguish purified channels: whether we exclude the use of PBS (\boxtimes), of polarisation flips (“negations” \ominus), or whether we allow both. This leads us to define the following equivalence relations:

► **Definition 13** (Observational equivalences). *Given two purified \mathcal{H} -channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$, we consider the three following refinements of observational equivalences (for $i \in \{0, 1, 2\}$): $[U, |\varepsilon\rangle, \mathcal{E}] \approx_i [U', |\varepsilon'\rangle, \mathcal{E}']$ if $\forall C[\cdot] \in \mathcal{C}_i$, $\llbracket C[U, |\varepsilon\rangle, \mathcal{E}] \rrbracket = \llbracket C[U', |\varepsilon'\rangle, \mathcal{E}'] \rrbracket$, where:*

- \mathcal{C}_0 is the set of \boxtimes -free contexts $C[\cdot] : \mathcal{H}^{(1)}$;
- \mathcal{C}_1 is the set of \ominus -free contexts $C[\cdot] : \mathcal{H}^{(1)}$;
- \mathcal{C}_2 is the set of all contexts $C[\cdot] : \mathcal{H}^{(1)}$.

Note that contexts in \mathcal{C}_0 do not perform any coherent control; these consist in just a linear sequence of gates and negations, possibly composed in parallel with closed loops (i.e., traces of such sequences), including a hole gate somewhere. It is clear, by deformation of diagrams, that more general contexts can always be described as follows:

► **Proposition 14.** *For any context $C[\cdot] \in \mathcal{C}_2$ there exists an extended PBS-diagram D such that $C[\cdot] = \boxed{D, \square}$. Moreover if $C[\cdot] \in \mathcal{C}_1$ then D can be chosen \ominus -free.*

► **Remark 15.** In Definition 13 we only consider contexts with a single input/output wire. This is because we intend to use contexts to distinguish purified channels; now, if one can distinguish two purified channels with a context of type $\mathcal{H}^{(n)}$ but no context of type $\mathcal{H}^{(1)}$, then intuitively this means that the extra power comes from the preparation of the initial state and/or some particular measurement, which are not represented in the context. Actually, except in the \mathcal{C}_0 case, allowing multiple input/output wires does not increase the distinguishability power of the contexts.

3.2 Observational equivalence using PBS-free contexts

Let us start by characterising which purified channels are indistinguishable by \boxtimes -free contexts in \mathcal{C}_0 . Not surprisingly, we recover the usual indistinguishability by input/output tests, which is captured by the fact that the two purified channels lead to the same superoperator:⁸

► **Definition 16** ((First-level) Superoperator). *Given a purified \mathcal{H} -channel $[U, |\varepsilon\rangle, \mathcal{E}]$, let $\mathcal{S}_{[U, |\varepsilon\rangle, \mathcal{E}]}^{(1)} : \mathcal{L}(\mathcal{H}) \rightarrow \mathcal{L}(\mathcal{H}) = \rho \mapsto \text{Tr}_{\mathcal{E}}(U(\rho \otimes |\varepsilon\rangle\langle\varepsilon|)U^\dagger)$ be the (“first-level”) superoperator of $[U, |\varepsilon\rangle, \mathcal{E}]$. Graphically,*

$$\mathcal{S}_{[U, |\varepsilon\rangle, \mathcal{E}]}^{(1)} := \begin{array}{c} \text{---} \\ \text{---} \end{array} \boxed{U} \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array}$$

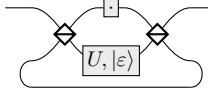
⁸ In other words, if two purified channels can be distinguished using a \boxtimes -free context, then they could already be distinguished with simply an input/output test (or with a trivial context \square).

► **Theorem 17.** *Given two purified \mathcal{H} -channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$, $[U, |\varepsilon\rangle, \mathcal{E}] \approx_0 [U', |\varepsilon'\rangle, \mathcal{E}']$ iff they have the same (first-level) superoperator. Graphically,*

$$[U, |\varepsilon\rangle, \mathcal{E}] \approx_0 [U', |\varepsilon'\rangle, \mathcal{E}'] \quad \text{iff} \quad \begin{array}{c} \text{---} \\ | \varepsilon \rangle \text{---} \boxed{U} \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ | \varepsilon' \rangle \text{---} \boxed{U'} \text{---} \\ \text{---} \end{array} \quad (\text{S1})$$

3.3 Observational equivalence using negation-free contexts

Allowing contexts with PBS significantly increases their power to distinguish purified channels. In [1], a particular kind of coherent control – namely, the “*first half of a quantum switch*” [6, 2, 12] – has been considered, which can be rephrased using contexts of the form:



The authors proved that with these particular contexts, two purified channels leading to the same (first-level) superoperator are indistinguishable if and only if they also have the same (first-level) transformation matrix, which is defined as follows⁹

► **Definition 18** ((First-level) Transformation Matrix). *Given a purified \mathcal{H} -channel $[U, |\varepsilon\rangle, \mathcal{E}]$, let $T_{[U, |\varepsilon\rangle, \mathcal{E}]}^{(1)} := (I_{\mathcal{H}} \otimes \langle \varepsilon |) U (I_{\mathcal{H}} \otimes |\varepsilon \rangle) \in \mathcal{L}(\mathcal{H})$ be the (“first-level”) transformation matrix of $[U, |\varepsilon\rangle, \mathcal{E}]$. Graphically,*

$$T_{[U, |\varepsilon\rangle, \mathcal{E}]}^{(1)} := \begin{array}{c} \text{---} \\ | \varepsilon \rangle \text{---} \boxed{U} \text{---} \\ \langle \varepsilon | \end{array}$$

We extend this result to any $\text{---} \ominus \text{---}$ -free context.

► **Theorem 19.** *Given two purified \mathcal{H} -channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$, $[U, |\varepsilon\rangle, \mathcal{E}] \approx_1 [U', |\varepsilon'\rangle, \mathcal{E}']$ iff they have the same (first-level) superoperator and the same (first-level) transformation matrix. Graphically,*

$$[U, |\varepsilon\rangle, \mathcal{E}] \approx_1 [U', |\varepsilon'\rangle, \mathcal{E}'] \quad \text{iff} \quad \left\{ \begin{array}{l} \begin{array}{c} \text{---} \\ | \varepsilon \rangle \text{---} \boxed{U} \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ | \varepsilon' \rangle \text{---} \boxed{U'} \text{---} \\ \text{---} \end{array} \quad (\text{S1}) \\ \begin{array}{c} \text{---} \\ | \varepsilon \rangle \text{---} \boxed{U} \text{---} \\ \langle \varepsilon | \end{array} = \begin{array}{c} \text{---} \\ | \varepsilon' \rangle \text{---} \boxed{U'} \text{---} \\ \langle \varepsilon' | \end{array} \quad (\text{T1}) \end{array} \right.$$

One can illustrate how the transformation matrices enter the game by considering for example the following context :

By plugging in $[U, |\varepsilon\rangle, \mathcal{E}]$, the extended PBS-diagram maps a pure input state $\frac{|\rightarrow\rangle + |\uparrow\rangle}{\sqrt{2}} \otimes |\psi\rangle \in \mathbb{C}^{\{\rightarrow, \uparrow\}} \otimes \mathcal{H}$ (together with the environment initial state $|\varepsilon\rangle \in \mathcal{E}$) to the state $\frac{1}{\sqrt{2}}|\rightarrow\rangle \otimes |\psi\rangle \otimes |\varepsilon\rangle + \frac{1}{\sqrt{2}}|\uparrow\rangle \otimes U(|\psi\rangle \otimes |\varepsilon\rangle)$, so that after tracing out the environment a cross term $\frac{1}{2}|\uparrow\rangle\langle\rightarrow| \otimes \text{Tr}_{\mathcal{E}}[U(|\psi\rangle\langle\psi| \otimes |\varepsilon\rangle\langle\varepsilon|)] = \frac{1}{2}|\uparrow\rangle\langle\rightarrow| \otimes T_{[U, |\varepsilon\rangle, \mathcal{E}]}^{(1)}|\psi\rangle\langle\psi|$ appears.

⁹ Originally, in [1], the transformation matrix was defined for a given unitary purification of a CPTP map $S : \mathcal{L}(\mathcal{H}) \rightarrow \mathcal{L}(\mathcal{H})$ in the form $U : |\psi\rangle_{\mathcal{H}} \otimes |\varepsilon\rangle \mapsto \sum_i K_i |\psi\rangle_{\mathcal{H}} \otimes |i\rangle_{\mathcal{E}}$ (where the K_i 's are Kraus operators of S , and where an environment space \mathcal{E} was introduced, with an orthonormal basis $\{|i\rangle_{\mathcal{E}}\}_i$ and an initial state $|\varepsilon\rangle$), as $T := \sum_i \langle \varepsilon | i \rangle_{\mathcal{E}} K_i$. This is indeed consistent with our Definition 18 here, as with these notations $U(I_{\mathcal{H}} \otimes |\varepsilon\rangle) = \sum_i K_i \otimes |i\rangle_{\mathcal{E}}$, so that $(I_{\mathcal{H}} \otimes \langle \varepsilon |) U (I_{\mathcal{H}} \otimes |\varepsilon\rangle) = \sum_i \langle \varepsilon | i \rangle_{\mathcal{E}} K_i = T$.

We note also that the two conditions (S1) and (T1) are nonredundant, i.e., one does not imply the other. Indeed, there exist cases where $\mathcal{S}_{[U,|\varepsilon],\mathcal{E}]}^{(1)} = \mathcal{S}_{[U',|\varepsilon'],\mathcal{E}']}^{(1)}$ but $T_{[U,|\varepsilon],\mathcal{E}]}^{(1)} \neq T_{[U',|\varepsilon'],\mathcal{E}']}^{(1)}$ (e.g., given any \mathcal{H} , $\mathcal{E} = \mathcal{E}' = \mathbb{C}$, $U = I_{\mathcal{H}}$, $U' = -I_{\mathcal{H}}$ and $|\varepsilon\rangle = |\varepsilon'\rangle = |1\rangle$), and cases where $\mathcal{S}_{[U,|\varepsilon],\mathcal{E}]}^{(1)} \neq \mathcal{S}_{[U',|\varepsilon'],\mathcal{E}']}^{(1)}$ but $T_{[U,|\varepsilon],\mathcal{E}]}^{(1)} = T_{[U',|\varepsilon'],\mathcal{E}']}^{(1)}$ (e.g., $\mathcal{H} = \mathcal{E} = \mathcal{E}' = \mathbb{C}^2$, $U = I_{\mathcal{H}} \otimes X$, $U' = X \otimes X$ and $|\varepsilon\rangle = |\varepsilon'\rangle = |0\rangle$).¹⁰

3.4 Observational equivalence using general contexts

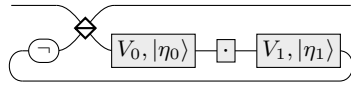
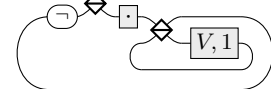
We will now see that allowing negations (\neg) increases the power of contexts to distinguish purified channels. To characterise the indistinguishability of purified channels with arbitrary contexts, we introduce *second-level* superoperators and *second-level* transformation matrices:

► **Definition 20** (Second-level Superoperator and Transformation Matrix). *Given a purified \mathcal{H} -channel $[U, |\varepsilon], \mathcal{E}]$, let $\mathcal{S}_{[U,|\varepsilon],\mathcal{E}]}^{(2)} : \mathcal{L}(\mathcal{H}^{\otimes 2}) \rightarrow \mathcal{L}(\mathcal{H}^{\otimes 2}) = \rho \mapsto \text{Tr}_{\mathcal{E}}(U^{(2)}(\rho \otimes |\varepsilon\rangle\langle\varepsilon|)U^{(2)\dagger})$ be the “second-level” superoperator and $T_{[U,|\varepsilon],\mathcal{E}]}^{(2)} := (I_{\mathcal{H}^{\otimes 2}} \otimes \langle\varepsilon|)U^{(2)}(I_{\mathcal{H}^{\otimes 2}} \otimes |\varepsilon\rangle) \in \mathcal{L}(\mathcal{H}^{\otimes 2})$ be the “second-level” transformation matrix of $[U, |\varepsilon], \mathcal{E}]$, where $U^{(2)} := (I_{\mathcal{H}} \otimes U)(\mathfrak{S} \otimes I_{\mathcal{E}})(I_{\mathcal{H}} \otimes U)$ and $\mathfrak{S} := |\psi_1\rangle \otimes |\psi_2\rangle \mapsto |\psi_2\rangle \otimes |\psi_1\rangle$ is the swap operator. Graphically, $U^{(2)} = \overline{\boxed{U}} \overline{\boxed{U}}$,*

$$\mathcal{S}_{[U,|\varepsilon],\mathcal{E}]}^{(2)} := \overline{\boxed{U}} \overline{\boxed{U}} \quad \text{and} \quad T_{[U,|\varepsilon],\mathcal{E}]}^{(2)} := \langle\varepsilon| \overline{\boxed{U}} \overline{\boxed{U}} |\varepsilon\rangle$$

► **Theorem 21.** *Given two purified \mathcal{H} -channels $[U, |\varepsilon], \mathcal{E}]$ and $[U', |\varepsilon'], \mathcal{E}']$, $[U, |\varepsilon], \mathcal{E}] \approx_2 [U', |\varepsilon'], \mathcal{E}']$ iff they have the same (first level) transformation matrix, the same second level superoperator and the same second level transformation matrix. Graphically,*

$$[U, |\varepsilon], \mathcal{E}] \approx_2 [U', |\varepsilon'], \mathcal{E}'] \quad \text{iff} \quad \left\{ \begin{array}{l} |\varepsilon\rangle \overline{\boxed{U}} \langle\varepsilon| = |\varepsilon'\rangle \overline{\boxed{U'}} \langle\varepsilon'| \quad (\text{T1}) \\ |\varepsilon\rangle \overline{\boxed{U}} \overline{\boxed{U}} \langle\varepsilon| = |\varepsilon'\rangle \overline{\boxed{U'}} \overline{\boxed{U'}} \langle\varepsilon'| \quad (\text{S2}) \\ |\varepsilon\rangle \overline{\boxed{U}} \overline{\boxed{U}} \langle\varepsilon| = |\varepsilon'\rangle \overline{\boxed{U'}} \overline{\boxed{U'}} \langle\varepsilon'| \quad (\text{T2}) \end{array} \right.$$

The contexts used in the proof to show that the constraints (S2) and (T2) are required are of the form  and , respectively, for some specific choices of purified channels $[V_0, |\eta_0\rangle, \mathcal{H} \otimes \mathbb{C}^2]$, $[V_1, |\eta_1\rangle, \mathcal{H} \otimes \mathbb{C}^2]$ and $[V, 1, \mathbb{C}]$. Hence, if either the second level superoperators or the second level transformation matrices of two purified channels differ, then the channels can be distinguished by using such contexts.

One may have expected the condition (S1) – i.e., that the two channels have the same first-level superoperator – to also appear in Theorem 21 (as it did in the previous two cases). This would however have been redundant, as can be seen from the following remark:

¹⁰ Where $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

► Remark 22. Two purified channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$ having the same second level superoperator also have the same first level superoperator, i.e., Condition (S2) implies (S1).

We note, on the other hand, that the three remaining conditions (T1), (S2) and (T2) are nonredundant. I.e., for each of the three there exist cases where only this condition is not satisfied, and where $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$ can be distinguished. E.g., with $\mathcal{E} = \mathcal{E}' = \mathbb{C}$, $U = I_{\mathcal{H}}, U' = -I_{\mathcal{H}}, |\varepsilon\rangle = |\varepsilon'\rangle = 1$, only (T1) fails to hold; with $\mathcal{H} = \mathcal{E} = \mathcal{E}' = \mathbb{C}^2$, $U = \text{CNOT}, U' = (\sqrt{Z} \otimes Z)\text{CNOT}, |\varepsilon\rangle = |\varepsilon'\rangle = |0\rangle$, only (S2) fails to hold; and with $\mathcal{H} = \mathcal{E} = \mathcal{E}' = \mathbb{C}^2$, $U = I_{\mathcal{H}} \otimes X, U' = I_{\mathcal{H}} \otimes ZX, |\varepsilon\rangle = |\varepsilon'\rangle = |0\rangle$, only (T2) fails to be satisfied.¹¹

4 Observational equivalence beyond PBS-diagrams

In this section, we define a new equivalence relation, inspired by the uniqueness (up to an isometry) of Stinespring's dilations, which subsumes the observational equivalences defined so far. For that let us first introduce an isometry-based preorder over purified channels:

► **Definition 23.** Given two purified \mathcal{H} -channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$, one has $[U, |\varepsilon\rangle, \mathcal{E}] \triangleleft_{iso} [U', |\varepsilon'\rangle, \mathcal{E}']$ if there exists an isometry $W: \mathcal{E} \rightarrow \mathcal{E}'$ s.t. $W|\varepsilon\rangle = |\varepsilon'\rangle$ and $(I_{\mathcal{H}} \otimes W)U = U'(I_{\mathcal{H}} \otimes W)$. In pictures:

$$|\varepsilon\rangle \text{---} \boxed{W} \text{---} = |\varepsilon'\rangle \quad \begin{array}{c} \mathcal{H} \\ \text{---} \\ \boxed{U} \\ \text{---} \\ \mathcal{E} \end{array} \text{---} \begin{array}{c} \mathcal{H} \\ \text{---} \\ \boxed{W} \\ \text{---} \\ \mathcal{E}' \end{array} = \begin{array}{c} \mathcal{H} \\ \text{---} \\ \boxed{W} \\ \text{---} \\ \mathcal{E}' \end{array} \text{---} \begin{array}{c} \mathcal{H} \\ \text{---} \\ \boxed{U'} \\ \text{---} \\ \mathcal{E}' \end{array}$$

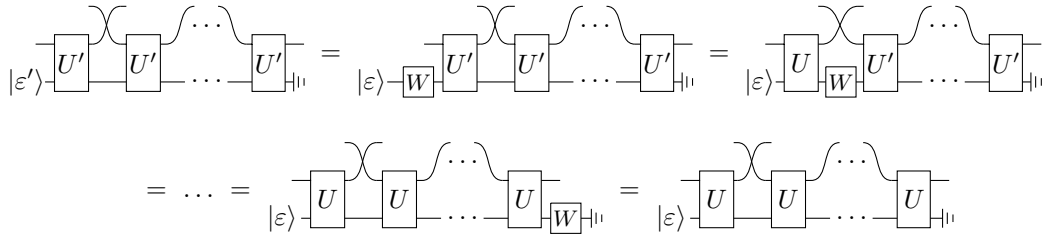
Note that \triangleleft_{iso} is not an equivalence relation. It is not symmetric; moreover, its symmetric closure is not transitive.¹² This leads us to consider the following:

► **Definition 24 (Iso-equivalence).** The iso-equivalence of purified channels is defined as the symmetric and transitive closure of \triangleleft_{iso} : $\approx_{iso} := \triangleleft_{iso}^*$.

The iso-equivalence is a candidate for characterising indistinguishability of purified channels in more general coherent-control settings. Actually, if $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$ are two iso-equivalent purified channels, then intuitively, in any coherent-control setting, $[U, |\varepsilon\rangle, \mathcal{E}]$ can be replaced by $[U', |\varepsilon'\rangle, \mathcal{E}']$ without changing the global behaviour. Indeed, the evolution of the environment associated with the purified channel is roughly speaking the same (up to the isometry W): initialised in the state $W|\varepsilon\rangle$ (and with the data register in the state $|\phi\rangle$), the application of U' leads to the state $U'(I_{\mathcal{H}} \otimes W)(|\phi\rangle \otimes |\varepsilon\rangle)$, which is equal to $(I_{\mathcal{H}} \otimes W)U(|\phi\rangle \otimes |\varepsilon\rangle)$. So applying U' somehow first cancels the application of W , then applies U , and finally applies W again – which will be cancelled again by the next application of U' , and so on. The last application of W is absorbed when the environment is traced out. In pictures:

¹¹ Where $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, $\sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ and $\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$.

¹² Taking $\mathcal{H} = \mathbb{C}$, one has $[1, 1, \mathbb{C}] \triangleleft_{iso} [I_{\mathbb{C}^2}, |0\rangle, \mathbb{C}^2]$ (with $W = |0\rangle$) but $\neg([I_{\mathbb{C}^2}, |0\rangle, \mathbb{C}^2] \triangleleft_{iso} [1, 1, \mathbb{C}])$ (as there is no isometry from \mathbb{C}^2 to \mathbb{C}). With the Pauli operator $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ one also has $[1, 1, \mathbb{C}] \triangleleft_{iso} [Z, |0\rangle, \mathbb{C}^2]$ (again with $W = |0\rangle$), but $[I_{\mathbb{C}^2}, |0\rangle, \mathbb{C}^2]$ and $[Z, |0\rangle, \mathbb{C}^2]$ are not in relation since there is no unitary W such that $WI_{\mathbb{C}^2} = ZW$ (as $I_{\mathbb{C}^2}$ and Z have distinct eigenvalues).



In the framework of PBS-diagrams, one can actually show that the iso-equivalence subsumes, but does not coincide with the \approx_2 -equivalence (which in turn subsumes the \approx_1 - and \approx_0 -equivalences).

► **Proposition 25.** $\approx_{iso} \subsetneq \approx_2 \subsetneq \approx_1 \subsetneq \approx_0$.

Proof. [$\approx_{iso} \subseteq \approx_2$] Since \approx_2 is an equivalence relation it is enough to show that $\triangleleft_{iso} \subseteq \approx_2$. If $[U, |\varepsilon\rangle, \mathcal{E}] \triangleleft_{iso} [U', |\varepsilon'\rangle, \mathcal{E}']$, then the three conditions of Theorem 21 are satisfied, implying $[U, |\varepsilon\rangle, \mathcal{E}] \approx_2 [U', |\varepsilon'\rangle, \mathcal{E}']$.

[$\approx_2 \neq \approx_{iso}$] We consider the following two purified \mathbb{C} -channels: $[X, |0\rangle, \mathbb{C}^3]$ and $[XN, |0\rangle, \mathbb{C}^3]$ where $X = |x\rangle \mapsto |x-1 \bmod 3\rangle$ and $N = |x\rangle \mapsto (-1)^x|x\rangle$ are two (qutrit) unitary transformations. The two purified channels are \approx_2 -equivalent as they satisfy the conditions of Theorem 21. In order to show that they are not iso-equivalent, note that if two purified \mathbb{C} -channels $[U, |\varepsilon\rangle, \mathcal{E}]$ and $[U', |\varepsilon'\rangle, \mathcal{E}']$ are iso-equivalent then for any $k \geq 0$ one has $\langle \varepsilon | U^k | \varepsilon \rangle = \langle \varepsilon' | W U^k | \varepsilon \rangle = \langle \varepsilon' | U'^k W | \varepsilon \rangle = \langle \varepsilon' | U'^k | \varepsilon' \rangle$. Since $\langle 0 | X^3 | 0 \rangle = 1 \neq -1 = \langle 0 | (XN)^3 | 0 \rangle$, it follows that $[X, |0\rangle, \mathbb{C}]$ and $[XN, |0\rangle, \mathbb{C}]$ are indeed not iso-equivalent.

[$\approx_2 \subsetneq \approx_1 \subsetneq \approx_0$] The inclusions are clear from the characterisations of Theorems 17, 19 and 21, together with Remark 22. The fact that the inclusions are strict follows from the observations that the various conditions appearing in these theorems are non-redundant. ◀

Although for PBS-diagrams, the \approx_2 -equivalence characterises the observational equivalence of purified channels, it could thus be that more general coherent-control settings may distinguish \approx_2 -equivalent channels. For instance one can imagine including nonpolarising beam splitters, or more general rotations of the polarisation than just the negation, or even settings with “higher-dimensional polarisations”, which would allow a particle to go more than twice through each gate. Such a setting would be able for instance to distinguish the pair of purified channels used in the proof of Proposition 25.

We conjecture that two purified channels are not iso-equivalent if and only if they can be distinguished by some coherently-controlled quantum computation. Here, the notion of coherently-controlled quantum computation is left loosely defined, and corresponds intuitively to some generalisation of PBS-diagrams allowing a particle to go through a gate an arbitrary number of times.

5 Discussion

In this work, we have extended the PBS-diagrams framework of [8] to allow for the coherent control of more general quantum channels, described as purified channels. By defining observational equivalence relations, we have characterised which purified channels are distinguishable depending on the class of contexts allowed (defined as PBS-diagrams with a hole). We also proposed the more refined iso-equivalence, which appears as a candidate for channel indistinguishability in more general coherent-control setups than PBS-diagrams.

However, unlike the previous equivalence relations that can be verified with simple criteria – by comparing superoperators and transformation matrices – the iso-equivalence, defined as a transitive closure, is *a priori* not as easy to check in general.

The framework of PBS-diagrams considered here has a number of limitations, which could be lifted in future works. For instance, it would be of practical interest to allow for nonpolarising beam splitters and more general operations on the polarisation; to consider using higher-dimensional control systems, with generalised PBS; or to consider several particles going through the diagrams, possibly correlating the different local environments for future uses of the diagrams, and/or inducing interference effects. We note also that in our description of purified channels, the state of the environment does not evolve by itself, except when the flying particle goes through the channel and the unitary U is applied to the joint system. In fact, as long as each channel is used at most twice (as it was case in this paper), any free evolution of the environment between two uses could be included in U ; however, introducing such an evolution could make a difference if the channels are used more than twice, and the evolution is different between different uses.

Other open questions raised by our work here include equipping extended PBS-diagrams with an equational theory, as was done in [8] for the case of “pure” PBS-diagrams; lifting our observational equivalences to the diagrams themselves; and investigating more general coherent-control settings, to check in particular whether our iso-equivalence is indeed the good definition for general distinguishability, and if it has a more operational characterisation.

References

- 1 Alastair A. Abbott, Julian Wechs, Dominic Horsman, Mehdi Mhalla, and Cyril Branciard. Communication through coherent control of quantum channels. *Quantum*, 4:333, 2020. doi:10.22331/q-2020-09-24-333.
- 2 Mateus Araújo, Fabio Costa, and Časlav Brukner. Computational Advantage from Quantum-Controlled Ordering of Gates. *Physical Review Letters*, 113(25):250402, 2014. doi:10.1103/PhysRevLett.113.250402.
- 3 Cyril Branciard, Alexandre Clément, Mehdi Mhalla, and Simon Perdrix. Coherent control and distinguishability of quantum channels via PBS-diagrams, 2021. arXiv:2103.02073.
- 4 Titouan Carette, Simon Perdrix, Renaud Vilmart, and Emmanuel Jeandel. Completeness of Graphical Languages for Mixed States Quantum Mechanics. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Patras, Greece, 2019. doi:10.4230/LIPIcs.ICALP.2019.108.
- 5 Giulio Chiribella. Perfect discrimination of no-signalling channels via quantum superposition of causal structures. *Physical Review A*, 86(4):040301, 2012. doi:10.1103/PhysRevA.86.040301.
- 6 Giulio Chiribella, Giacomo Mauro D’Ariano, Paolo Perinotti, and Benoît Valiron. Quantum computations without definite causal structure. *Physical Review A*, 88:022318, August 2013. doi:10.1103/PhysRevA.88.022318.
- 7 Giulio Chiribella and Hlér Kristjánsson. Quantum shannon theory with superpositions of trajectories. *Proceedings of the Royal Society A*, 475:20180903, 2019. doi:10.1098/rspa.2018.0903.
- 8 Alexandre Clément and Simon Perdrix. PBS-calculus: A graphical language for coherent control of quantum computations. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.24.
- 9 Bob Coecke and Simon Perdrix. Environment and classical channels in categorical quantum mechanics. *Logical Methods in Computer Science*, Volume 8, Issue 4, November 2012. doi:10.2168/LMCS-8(4:14)2012.

- 10 Nicolai Friis, Vedran Dunjko, Wolfgang Dür, and Hans J. Briegel. Implementing quantum control for unknown subroutines. *Physical Review A*, 89:030303(R), 2014. doi:10.1103/PhysRevA.89.030303.
- 11 Nicolas Gisin, Noah Linden, Serge Massar, and Sandu Popescu. Error filtration and entanglement purification for quantum communication. *Physical Review A*, 72:012338, July 2005. doi:10.1103/PhysRevA.72.012338.
- 12 Kaumudibikash Goswami, Christina Giarmatzi, Michael Kewming, Fabio Costa, Cyril Branciard, Jacqueline Romero, and Andrew G. White. Indefinite causal order in a quantum switch. *Physical Review Letters*, 121:090503, August 2018. doi:10.1103/PhysRevLett.121.090503.
- 13 Hlér Kristjánsson, Giulio Chiribella, Sina Salek, Daniel Ebler, and Matthew Wilson. Resource theories of communication. *New Journal of Physics*, 22(7):073014, 2020. doi:10.1088/1367-2630/ab8ef7.
- 14 Saunders Mac Lane. Categorical algebra. *Bulletin of the American Mathematical Society*, 71:40–106, 1965. doi:10.1090/S0002-9904-1965-11234-4.
- 15 Daniel K. L. Oi. Interference of quantum channels. *Physical Review Letters*, 91:067902, 2003. doi:10.1103/PhysRevLett.91.067902.
- 16 Timothy M. Rambo, Joseph B. Altepeter, Prem Kumar, and Giacomo Mauro D’Ariano. Functional quantum computing: An optical approach. *Physical Review A*, 93:052321, May 2016. doi:10.1103/PhysRevA.93.052321.
- 17 W. Forrest Stinespring. Positive functions on C^* -algebras. *Proceedings of the American Mathematical Society*, 6:211–216, 1955. doi:10.1090/S0002-9939-1955-0069403-4.
- 18 Fabio Zanasi. *Interacting Hopf Algebras- the Theory of Linear Systems*. PhD thesis, Ecole normale supérieure de lyon - ENS LYON, 2015. arXiv:1805.03032.

A Structural congruence of PBS-diagrams

Bare PBS-diagrams, extended PBS-diagrams and contexts are defined up to the congruence generated by the following equalities (with all $n, m, k \geq 0$), where I_n is the “identity diagram” $I_n := \oplus^n(-)$ (graphically: $I_n = n\{\overline{\quad}\}$, with $I_0 = \{\overline{\quad}\}$); $\sigma_{1,n}$ is the “first-wire-goes-last diagram” defined inductively by $\sigma_{1,0} := -$ and $\sigma_{1,n+1} := (I_n \oplus \times) \circ (\sigma_{1,n} \oplus -)$ (graphically: $\sigma_{1,n} = n\{\overline{\quad}\}$); and $D : n$ denotes here either a bare PBS-diagram $D : n$, an extended PBS-diagram $D : \mathcal{H}^{(n)}$, or a context $C[\cdot] : \mathcal{H}^{(n)}$:

- *Neutrality of the identity:* for any $D : n$,

$$D \circ I_n = D = I_n \circ D$$

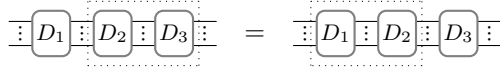
- *Neutrality of the empty diagram:* for any $D : n$,

$$\{\overline{\quad}\} \oplus D = D = D \oplus \{\overline{\quad}\}$$

22:16 Coherent Control and Distinguishability of Quantum Channels via PBS-Diagrams

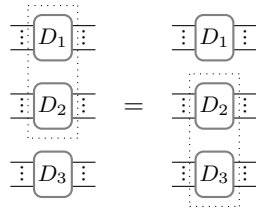
- *Associativity of the sequential composition:* for any $D_1, D_2, D_3 : n$,

$$(D_3 \circ D_2) \circ D_1 = D_3 \circ (D_2 \circ D_1)$$



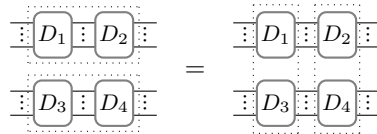
- *Associativity of the parallel composition:* for any $D_1 : n, D_2 : m$ and $D_3 : k$,

$$(D_1 \oplus D_2) \oplus D_3 = D_1 \oplus (D_2 \oplus D_3)$$



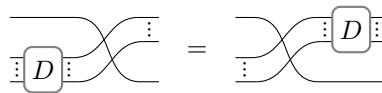
- *Compatibility of the sequential and parallel compositions:* for any $D_1, D_2 : n$ and $D_3, D_4 : m$,

$$(D_2 \circ D_1) \oplus (D_4 \circ D_3) = (D_2 \oplus D_4) \circ (D_1 \oplus D_3)$$



- *Naturality of the swap:* for any $D : n$,

$$\sigma_{1,n} \circ (- \oplus D) = (D \oplus -) \circ \sigma_{1,n}$$



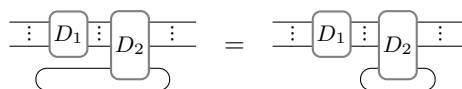
- *Inverse law:*

$$\bowtie \circ \bowtie = I_2$$

$$\text{crossing} = \text{parallel lines}$$

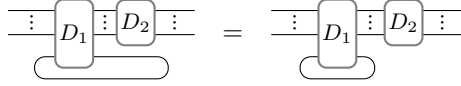
- *Naturality in the input:* for any $D_1 : n$ and $D_2 : n + 1$,

$$\text{Tr}(D_2 \circ (D_1 \oplus -)) = \text{Tr}(D_2) \circ D_1$$



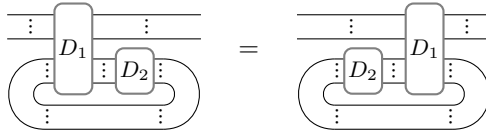
- *Naturality in the output:* for any $D_1 : n + 1$ and $D_2 : n$,

$$\text{Tr}((D_2 \oplus -) \circ D_1) = D_2 \circ \text{Tr}(D_1)$$



- *Dinaturality:* for any $D_1 : n + m$ and $D_2 : m$,

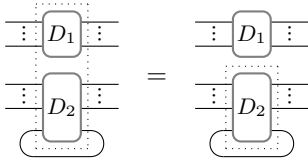
$$\text{Tr}^m((I_n \oplus D_2) \circ D_1) = \text{Tr}^m(D_1 \circ (I_n \oplus D_2))$$



where Tr^m denotes the m^{th} power of the trace operation.

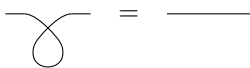
- *Superposing:* for any $D_1 : n$ and $D_2 : m + 1$,

$$\text{Tr}(D_1 \oplus D_2) = D_1 \oplus \text{Tr}(D_2)$$



- *Yanking:*

$$\text{Tr}(\infty) = -$$



These equalities are the coherence axioms of a traced PROP, that is, a PROP that is also a traced symmetric monoidal category. An explicit definition of the concept of traced PROP is given in [8]. See also [14] and [18] for a definition of PROPs and further details about them.

B Well-definedness of the word path semantics and compatibility with the structural congruence

It can be proved in the same way as for Propositions 5 and 6 in [8], that the word path semantics is well-defined despite the restriction that $k \leq 2$ in Rule (T_k) , that it is deterministic (i.e., that for any bare diagram $D : n$, polarisation $c \in \{\rightarrow, \uparrow\}$ and position $p \in [n]$, there exist some unique c', p' and w such that $(D, c, p) \xrightarrow{w} (c', p')$ – which allows us to define $c_{c,p}^D$, $p_{c,p}^D$ and $w_{c,p}^D$), and that conversely, for any target polarisation c' and position p' , there exist

c and p such that $(D, c, p) \xrightarrow{w} (c', p')$ for some w (in other words, the map $(c, p) \mapsto (c_{c,p}^D, p_{c,p}^D)$ is a bijection). We give here some additional details about the fact that it is invariant modulo diagram deformation:

► **Proposition 26.** *The word path semantics is invariant modulo diagram deformation.*

Proof. One has to check, for each of the equalities given in Appendix A, that the two sides have the same word path semantics. This is straightforward in each case except for dinaturality. In this case we first prove that Rule (\mathbb{T}_k^m) below follows from those of Definition 2:

$$\frac{D : n + m \quad \forall i \in \{0, \dots, k\}, (D, c_i, p_i) \xrightarrow{w_i} (c_{i+1}, p_{i+1}) \quad (p_{i+1} \geq n) \Leftrightarrow (i < k)}{(Tr^m(D), c_0, p_0) \xrightarrow{w_0 \dots w_k} (c_{k+1}, p_{k+1})} (\mathbb{T}_k^m)$$

for all $k, m \in \mathbb{N}$.

To prove this, we proceed by induction on m . The case $m = 0$ is trivial, and the case $m = 1$ corresponds to Rule (\mathbb{T}_k) of Definition 2 (the rule follows even for $k \geq 3$ since it is then not possible to satisfy its premises).

Now, assume that Rule (\mathbb{T}_k^m) follows from those of Definition 2. Let $D : n + m + 1$. Let $c_0 \in \{\rightarrow, \uparrow\}$ and $p_0 \in [n]$. Let $(c_1, p_1), \dots, (c_{k+1}, p_{k+1})$ be the (unique) sequence of couples such that $\forall i \in \{0, \dots, k\}, (D, c_i, p_i) \xrightarrow{w_i} (c_{i+1}, p_{i+1})$ and $(p_{i+1} \geq n) \Leftrightarrow (i < k)$ (that is, $k + 1$ is the first index after 0 such that $p_{k+1} < n$). Let $(c_{i_0}, p_{i_0}), \dots, (c_{i_{k'}+1}, p_{i_{k'}+1})$, with $0 = i_0 < i_1 < \dots < i_{k'} < i_{k'+1} = k + 1$, be the subsequence of $(c_1, p_1), \dots, (c_{k+1}, p_{k+1})$ where all couples with $p_i = n + m$ have been removed. For each $j \in \{0, \dots, k'\}$, by Rule (\mathbb{T}_k) one has $(Tr(D), c_{i_j}, p_{i_j}) \xrightarrow{w_{i_j} \dots w_{i_{j+1}-1}} (c_{i_{j+1}}, p_{i_{j+1}})$. Additionally, one has $Tr(D) : n + m$ and $(p_{i_{j+1}} \geq n) \Leftrightarrow (j < k')$, so that by Rule (\mathbb{T}_k^m) , one has $(Tr^{m+1}(D), c_0, p_0) \xrightarrow{w_0 \dots w_k} (c_{k+1}, p_{k+1})$, which validates Rule (\mathbb{T}_k^{m+1}) .

Given Rule (\mathbb{T}_k^m) for all k, m , we check the compatibility of the word path semantics with dinaturality as follows: given any $D_1 : n + m$ and $D_2 : m$ with $n, m \geq 0$, on the one hand one has

$$\left\{ \begin{array}{ll} ((I_n \oplus D_2) \circ D_1, c, p) \xrightarrow{w_{c,p}^{D_1}} (c_{c,p}^{D_1}, p_{c,p}^{D_1}) & \text{if } p_{c,p}^{D_1} < n \\ ((I_n \oplus D_2) \circ D_1, c, p) \xrightarrow{w_{c,p}^{D_1} w_{(c_{c,p}^{D_1}), (p_{c,p}^{D_1}-n)}^{D_2}} (c_{(c_{c,p}^{D_1}), (p_{c,p}^{D_1}-n)}^{D_2}, p_{(c_{c,p}^{D_1}), (p_{c,p}^{D_1}-n)}^{D_2}) + n & \text{if } p_{c,p}^{D_1} \geq n \end{array} \right.$$

so that given $c_0 \in \{\rightarrow, \uparrow\}$ and $p_0 \in [n]$, if one has a sequence $((I_n \oplus D_2) \circ D_1, c_0, p_0) \xrightarrow{w_0} (c_1, p_1), \dots, ((I_n \oplus D_2) \circ D_1, c_k, p_k) \xrightarrow{w_k} (c_{k+1}, p_{k+1})$ with $(p_{i+1} \geq n) \Leftrightarrow (i < k)$, then one has a sequence $(D_1, c_0, p_0) \xrightarrow{w'_0} (c'_1, p'_1), (D_2, c'_1, p'_1 - n) \xrightarrow{w''_1} (c_1, p_1 - n), (D_1, c_1, p_1) \xrightarrow{w'_1} (c'_1, p'_1), \dots, (D_1, c_{k-1}, p_{k-1}) \xrightarrow{w'_{k-1}} (c'_k, p'_k), (D_2, c'_k, p'_k - n) \xrightarrow{w''_k} (c_k, p_k - n), (D_1, c_k, p_k) \xrightarrow{w'_k} (c_{k+1}, p_{k+1})$ with $\forall i \in \{0, \dots, k-1\}, w'_i w''_{i+1} = w_i$, and $w'_k = w_k$, so that $(Tr^m((I_n \oplus D_2) \circ D_1), c_0, p_0) \xrightarrow{w'_0 w''_1 \dots w'_{k-1} w''_k w'_k} (c_{k+1}, p_{k+1})$.

On the other hand, one has

$$\left\{ \begin{array}{ll} (D_1 \circ (I_n \oplus D_2), c, p) \xrightarrow{w_{c,p}^{D_1}} (c_{c,p}^{D_1}, p_{c,p}^{D_1}) & \text{if } p < n \\ (D_1 \circ (I_n \oplus D_2), c, p) \xrightarrow{w_{c,p-n}^{D_2} w_{(c_{c,p-n}, (p_{c,p-n}+n))}^{D_1}} (c_{(c_{c,p-n}, (p_{c,p-n}+n))}^{D_1}, p_{(c_{c,p-n}, (p_{c,p-n}+n))}^{D_1}) & \text{if } p \geq n \end{array} \right.$$

so that given $c_0 \in \{\rightarrow, \uparrow\}$ and $p_0 \in [n]$, if one has a sequence $(D_1 \circ (I_n \oplus D_2), c_0, p_0) \xrightarrow{\tilde{w}_0} (c'_1, p'_1), \dots, (D_1 \circ (I_n \oplus D_2), c'_k, p'_k) \xrightarrow{\tilde{w}_k} (c'_{k+1}, p'_{k+1})$ with $(p_{i+1} \geq n) \Leftrightarrow (i < k)$, then one has a sequence $(D_1, c_0, p_0) \xrightarrow{w'_0} (c'_1, p'_1), (D_2, c'_1, p'_1 - n) \xrightarrow{w'_1} (c_1, p_1 - n), (D_1, c_1, p_1) \xrightarrow{w'_1} (c'_1, p'_1), \dots, (D_1, c_{k-1}, p_{k-1}) \xrightarrow{w'_{k-1}} (c'_k, p'_k), (D_2, c'_k, p'_k - n) \xrightarrow{w'_k} (c_k, p_k - n), (D_1, c_k, p_k) \xrightarrow{w'_k} (c_{k+1}, p_{k+1})$ with $w'_0 = \tilde{w}_0$ and $\forall i \in \{0, \dots, k-1\}, w'_i w'_i = \tilde{w}_i$, so that one has $(c'_{k+1}, p'_{k+1}) = (c_{k+1}, p_{k+1})$ and $(Tr^m(D_1 \circ (I_n \oplus D_2)), c_0, p_0) \xrightarrow{w'_0 w'_1 \dots w'_{k-1} w'_k w'_k} (c_{k+1}, p_{k+1})$. This proves that the two sides of the equality have the same semantics. ◀

C Circuit notations

In this paper, we further develop the graphical representation of coherent control by means of PBS-diagrams, but we also use circuit-like notations when it is convenient to represent sequential and parallel compositions of linear transformations $\mathcal{H}_{\text{in}} \rightarrow \mathcal{H}_{\text{out}}$ for some Hilbert spaces \mathcal{H}_{in} and \mathcal{H}_{out} (e.g., unitary operations, density matrices or matrices of the form $|i\rangle\langle j|$) and linear maps $\mathcal{L}(\mathcal{H}_{\text{in}}) \rightarrow \mathcal{L}(\mathcal{H}_{\text{out}})$ (i.e., superoperators). We briefly review these circuit-like notations: given a linear transformation $U : \mathcal{H}_1 \otimes \dots \otimes \mathcal{H}_n \rightarrow \mathcal{H}'_1 \otimes \dots \otimes \mathcal{H}'_k$,

$$\begin{array}{c} \mathcal{H}_1 \\ \vdots \\ \mathcal{H}_n \end{array} \boxed{U} \begin{array}{c} \mathcal{H}'_1 \\ \vdots \\ \mathcal{H}'_k \end{array}$$

is a circuit of type $\mathcal{H}_1 \otimes \dots \otimes \mathcal{H}_n \rightarrow \mathcal{H}'_1 \otimes \dots \otimes \mathcal{H}'_k$. Note that the Hilbert spaces on the wires are generally omitted when these are clear from the context.

The identity operator on a Hilbert space is represented as a wire. Sequential composition consists in plugging two circuits (with the appropriate types) in a row, and tensor product consists in putting two circuits in parallel, e.g., for any linear maps $U : \mathcal{H}_0 \rightarrow \mathcal{H}_1$, $V : \mathcal{H}_1 \rightarrow \mathcal{H}_2$, $W : \mathcal{H}_2 \rightarrow \mathcal{H}_3$:

$$\begin{array}{c} \mathcal{H}_0 \\ \boxed{U} \\ \mathcal{H}_1 \end{array} \begin{array}{c} \mathcal{H}_1 \\ \boxed{V} \\ \mathcal{H}_2 \end{array} = \begin{array}{c} \mathcal{H}_0 \\ \boxed{V \circ U} \\ \mathcal{H}_2 \end{array} \quad \begin{array}{c} \mathcal{H}_0 \\ \boxed{U} \\ \mathcal{H}_1 \end{array} \begin{array}{c} \mathcal{H}_2 \\ \boxed{W} \\ \mathcal{H}_3 \end{array} = \begin{array}{c} \mathcal{H}_0 \\ \boxed{U \otimes W} \\ \mathcal{H}_3 \end{array}$$

The associativity of both \circ and \otimes , and the mixed-product property $((U' \otimes V') \circ (U \otimes V)) = (U' \circ U) \otimes (V' \circ V)$ for some $U : \mathcal{H}_0 \rightarrow \mathcal{H}_1$, $U' : \mathcal{H}_1 \rightarrow \mathcal{H}_2$, $V : \mathcal{H}_3 \rightarrow \mathcal{H}_4$, $V' : \mathcal{H}_4 \rightarrow \mathcal{H}_5$ guarantee the nonambiguity of the circuit-like notations. Quantum states (resp. their adjoints) can be added to input (resp. output) wires, e.g., $|\varphi\rangle \boxed{U} \langle\psi| = \langle\psi|U|\varphi\rangle$.

With the swap $\begin{array}{c} \mathcal{H}_1 \\ \mathcal{H}_2 \end{array} \begin{array}{c} \mathcal{H}_2 \\ \mathcal{H}_1 \end{array} = |\varphi_1\rangle \otimes |\varphi_2\rangle \mapsto |\varphi_2\rangle \otimes |\varphi_1\rangle$, and with quantum states $|\varphi\rangle \in \mathcal{H}$ (resp. their adjoints $\langle\psi| \in \mathcal{H}^\dagger$) seen as linear transformations $\mathbb{C} \rightarrow \mathcal{H}$ (resp. $\mathcal{H} \rightarrow \mathbb{C}$), circuits form a strict symmetric monoidal category. That is, in addition to the fact that the notation is not ambiguous, circuits can be deformed at will (as long as their topology is preserved) without changing the transformation that is represented.

Following [9, 4], we further extend these notations to represent linear maps $\mathcal{L}(\mathcal{H}_{\text{in}}) \rightarrow \mathcal{L}(\mathcal{H}_{\text{out}})$, using the “ground” symbol \dashv . Given a “pure” (i.e., \dashv -free) circuit, plugging one (or several) \dashv in its output wire(s) corresponds essentially to tracing out the corresponding systems – or more precisely, to defining the map that takes an operator (typically, a density matrix, ρ) acting on the input Hilbert spaces, applies the linear map defined by the circuit (as in $\rho \mapsto U\rho U^\dagger$), and traces out the systems to which the ground symbol is attached, e.g.,

$$\begin{array}{c} \mathcal{H}_1 \\ \mathcal{H}_2 \\ \mathcal{H}_3 \end{array} \left[\begin{array}{c} \mathcal{H}'_1 \\ U \\ \mathcal{H}'_2 \\ \mathcal{H}'_3 \end{array} \right]_{\perp} = \rho \mapsto \text{Tr}_{\mathcal{H}'_2 \otimes \mathcal{H}'_3} (U \rho U^\dagger)$$

$$\begin{array}{c} \mathcal{H}_0 \\ |\varphi\rangle \end{array} \left[\begin{array}{c} \mathcal{H}_1 \\ V \\ \varepsilon \end{array} \right]_{\perp} = \rho \mapsto \text{Tr}_{\mathcal{E}} (V(\rho \otimes |\varphi\rangle\langle\varphi|)V^\dagger)$$

where the top example defines a map $\mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3) \rightarrow \mathcal{L}(\mathcal{H}'_1)$, and the bottom example defines a map $\mathcal{L}(\mathcal{H}_0) \rightarrow \mathcal{L}(\mathcal{H}_1)$. We say that such circuits are of type $\mathcal{L}(\mathcal{H}_{\text{in}}) \rightarrow \mathcal{L}(\mathcal{H}_{\text{out}})$.

► **Remark 27.** With these definitions, for a circuit with input Hilbert spaces $\mathcal{H}_1, \dots, \mathcal{H}_n$ and output Hilbert spaces $\mathcal{H}'_1, \dots, \mathcal{H}'_k$ to represent a linear map $\mathcal{L}(\mathcal{H}_1 \otimes \dots \otimes \mathcal{H}_n) \rightarrow \mathcal{L}(\mathcal{H}'_1 \otimes \dots \otimes \mathcal{H}'_k)$, it must contain at least one \perp symbol. As a consequence the CPTP map $\rho \mapsto U \rho U^\dagger$ cannot be represented as \boxed{U} (which is a “pure” circuit) but for instance as \boxed{U} .

Note that one can consider \perp as a generator $\mathcal{L}(\mathcal{H}) \rightarrow \mathcal{L}(\mathbb{C}) = \mathbb{C}$ and place it anywhere in the circuit. Because of the strict symmetric monoidal structure of \perp -free circuits and the fact that $\bigcap_{\perp} = \perp$, this does not create ambiguity since all ways of pulling the \perp symbols to the right give the same linear map. Moreover, circuits with this additional generator still form a strict symmetric monoidal category.

Reconfiguring Independent Sets on Interval Graphs

Marcin Briański ✉

Theoretical Computer Science Department, Faculty of Mathematics and Computer Science,
Jagiellonian University, Kraków, Poland

Stefan Felsner ✉ 🏠

Institut für Mathematik, Technische Universität Berlin, Germany

Jędrzej Hodor ✉

Theoretical Computer Science Department, Faculty of Mathematics and Computer Science,
Jagiellonian University, Kraków, Poland

Piotr Micek ✉ 🏠

Theoretical Computer Science Department, Faculty of Mathematics and Computer Science,
Jagiellonian University, Kraków, Poland

Abstract

We study reconfiguration of independent sets in interval graphs under the token sliding rule. We show that if two independent sets of size k are reconfigurable in an n -vertex interval graph, then there is a reconfiguration sequence of length $\mathcal{O}(k \cdot n^2)$. We also provide a construction in which the shortest reconfiguration sequence is of length $\Omega(k^2 \cdot n)$.

As a counterpart to these results, we also establish that **Independent Set Reconfiguration** is PSPACE-hard on incomparability graphs, of which interval graphs are a special case.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases reconfiguration, independent sets, interval graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.23

Funding M. Briański, J. Hodor, P. Micek are partially supported by a Polish National Science Center grant (BEETHOVEN; UMO-2018/31/G/ST1/03718).

1 Introduction

Let G be a graph and let k be a positive integer. The *reconfiguration graph* $R_k(G)$ of independent sets of size k in G has as its vertex set the set of all independent sets of size k in G and two independent sets I, J of size k in G are adjacent in $R_k(G)$ whenever $I \Delta J = \{u, v\}$ and $uv \in E(G)$.

When two sets I and J are in the same component of $R_k(G)$ we say that I and J are *reconfigurable*. In such case, we can perform the following transformation process: (1) start by placing one token on each vertex of I ; (2) in each step move one of the tokens to a neighboring vertex in G but always keep the property that the vertices occupied by tokens induce an independent set in G ; (3) finish with tokens occupying all vertices of J . Let (I_0, \dots, I_m) be a path from I to J in $R_k(G)$ so $I_0 = I, I_m = J$. For $i \in \{1, \dots, m\}$, let (u_i, v_i) be a pair of vertices in G such that $I_{i-1} \setminus I_i = \{u_i\}$ and $I_i \setminus I_{i-1} = \{v_i\}$. We say that the sequence $((u_1, v_1), \dots, (u_m, v_m))$ is a *reconfiguration sequence* from I to J in G .

A large body of research is focused on the computational complexity of the corresponding decision problem – **Independent Set Reconfiguration**: given a graph G and two independent sets I and J , determine whether I and J are reconfigurable. If we do not assume anything about the input graph this problem is PSPACE-complete, thus it is natural to investigate how its complexity changes when input graphs are restricted to a particular class of graphs. Demaine et al. [4] showed that the problem can be solved in polynomial time on trees. Lokshtanov and Mouawad [8] proved that it remains PSPACE-complete on bipartite graphs.



© Marcin Briański, Stefan Felsner, Jędrzej Hodor, and Piotr Micek;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 23; pp. 23:1–23:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Hearn and Demaine [6] showed that it is PSPACE-complete on planar graphs. There are polynomial time algorithms for the class of cographs [7] as well as claw-free graphs [2]. For a general survey of problems related to reconfiguration see [9].

Bonamy and Bousquet [1] presented an algorithm that given an interval graph G and two independent sets of size k in G verifies if the independent sets are reconfigurable in $\mathcal{O}(n^3)$ time. Curiously, their proof does not say anything meaningful about the length of the reconfiguration sequence if the two independent sets are reconfigurable. Since there are at most $\binom{n}{k}$ independent sets of size k , we get a trivial bound on the length of the reconfiguration sequence, namely: $\mathcal{O}(n^k)$. The question whether every two reconfigurable independent sets in an interval graph are connected by a reconfiguration sequence of polynomial length of degree independent of k was raised by Bousquet [3], which we answer in the affirmative in Theorem 1.

1.1 Our results

In this paper we will be mainly considering interval graphs. Our main results are stated in Theorems 1 and 2 below.

► **Theorem 1.** *Let G be an n -vertex interval graph and k be a positive integer. Then every component of $R_k(G)$ has diameter in $\mathcal{O}(k \cdot n^2)$.*

Moreover, there is a polynomial time algorithm that given G , k , and two independent sets I, J of size k in G decides if I and J are reconfigurable and if so outputs a reconfiguration sequence connecting them of length $\mathcal{O}(k \cdot n^2)$.

A lower bound construction shows that the bound on the length of a reconfiguration sequence given in Theorem 1 is close to tight.

► **Theorem 2.** *For all integers $m \geq 1$ and $k \geq 1$, there is an interval graph $G_{m,k}$ with $|V(G_{m,k})|$ in $\mathcal{O}(m+k)$ and two reconfigurable independent sets I, J of size k in $G_{m,k}$ such that every reconfiguration sequence connecting I and J in G is of length $\Omega(k^2 \cdot m)$.*

In light of Theorem 2, so long as $k \in \Theta(m)$, the bound of Theorem 1 is asymptotically tight. However, if k is small compared to n , the bounds of Theorems 1 and 2 can differ by a factor of $\mathcal{O}(n)$. This leads to an interesting problem in its own right. The authors are not aware of any example giving a superlinear lower bound on the length of a reconfiguration sequence when the number of tokens is constant – *even on the class of all graphs*. Specifically, the case $k = 2$ remains open.

Interval graphs are a special case in the more general class of incomparability graphs. As a somewhat expected (and indeed easy) result we have obtained the following theorem.

► **Theorem 3.** *Independent Set Reconfiguration is PSPACE-hard on incomparability graphs, even on incomparability graphs of posets of width at most w for some constant $w \in \mathbb{N}$.*

Another interesting specialization of the incomparability graphs are *permutation graphs*. In [5] the authors show a polynomial time algorithm solving Independent Set Reconfiguration on bipartite permutation graphs. We suspect that general permutation graphs should be amenable to methods similar to the tools we use here on interval graphs, however we have not been able to successfully apply them.

► **Conjecture 4.** *Independent Set Reconfiguration is solvable in polynomial time when restricted to permutation graphs.*

2 Preliminaries

A *component* of a graph G is a non-empty induced subgraph of G that is connected and is vertex-maximal under these properties. The *length* of a path is the number of edges in the path. The distance of two vertices u, v in a graph G is the minimum length of a path connecting u and v in G . The *diameter* of a connected graph is the maximum distance between any pair of vertices in the graph.

A graph G is an *interval graph* if the vertices of G can be associated with intervals on the real line in such a way that two vertices are adjacent in G if and only if the corresponding intervals intersect. Given an interval graph we will always fix an interval representation and identify the vertices with the corresponding intervals. We will also assume that all the endpoints of intervals in the representation are pairwise distinct. This can be easily achieved by perturbing the endpoints where it is needed.

Let S be a reconfiguration sequence from I to J in G . We sometimes say that we *apply* S to I in G and obtain an independent set $S(I)$. When $S = ((u, v))$ we also simply say that we *apply* a pair (u, v) to I and obtain $S(I) = (I \setminus \{u\}) \cup \{v\}$.

Given a sequence S , we denote by $S|_t$ the prefix of S of length t (so $S|_0$ is the empty sequence). Given a reconfiguration sequence $S = ((u_1, v_1), \dots, (u_m, v_m))$ from I to J in G , clearly for each $t \in \{0, \dots, m\}$, the prefix $S|_t$ is a reconfiguration sequence from I to the set $S|_t(I)$ in G . Thus, we also have $S|_0(I) = I$.

3 Upper bound: The Algorithm

In this section we are going to prove Theorem 1. For brevity, we omit implementation details as well as running time analyses of the algorithms outlined in this section. We note that a straightforward implementation of Algorithm 3 below (which is the procedure whose existence implies Theorem 1) would yield a $\mathcal{O}(n^3)$ algorithm.

Let G be an n -vertex interval graph and let k be a positive integer. We fix an interval representation of G distinguishing all the endpoints. There are two natural linear orderings on the vertices of G : \leq_{left} the order increasing along the left endpoints of the intervals and \leq_{right} the order increasing along the right endpoints of the intervals.

An independent set in G is a set of pairwise disjoint intervals, as such they are naturally ordered on the line. Thus given an independent set $A = \{a_1, \dots, a_\ell\}$ in G we will treat it as a tuple of intervals (a_1, \dots, a_ℓ) with $a_1 < \dots < a_\ell$ on the line. We define the projection $\pi_i(A) = a_i$ for each $i \in \{1, \dots, \ell\}$. Also when we apply (u, v) to an independent set A , we say that (u, v) *moves the i -th token* of A when $u = a_i$.

Let ℓ be a positive integer and let \mathcal{C} be a non-empty family of independent sets each of size ℓ in G . For $j \in \{1, \dots, \ell\}$ we define

$$\text{ex}_j(\mathcal{C}, \text{left}) = \min_{\leq_{\text{right}}} \{\pi_j(A) : A \in \mathcal{C}\}, \quad \text{ex}_j(\mathcal{C}, \text{right}) = \max_{\leq_{\text{left}}} \{\pi_j(A) : A \in \mathcal{C}\}.$$

For $p \in \{0, \dots, \ell\}$, we define the p -*extreme set* of \mathcal{C} to be

$$\bigcup_{1 \leq j \leq p} \{\text{ex}_j(\mathcal{C}, \text{left})\} \cup \bigcup_{p+1 \leq j \leq \ell} \{\text{ex}_j(\mathcal{C}, \text{right})\}.$$

We are going to show (Lemma 7) that every component \mathcal{C} of $R_k(G)$ contains all its p -extreme sets for $p \in \{0, \dots, k\}$. This gives a foundation for our algorithm: given two independent sets I and J as the input, we are going to devise two reconfiguration sequences,

23:4 Reconfiguring Independent Sets on Interval Graphs

transforming I into the $(k-1)$ -extreme set of its component in $R_k(G)$ and J into the $(k-1)$ -extreme set of its component in $R_k(G)$ respectively. As the $(k-1)$ -extreme set is a function of a connected component, we conclude that I and J are reconfigurable if and only if the obtained $(k-1)$ -extreme sets are equal. Thus, the essence of our work is to show that every independent set I can be reconfigured in $\mathcal{O}(k \cdot n^2)$ steps into the $(k-1)$ -extreme set of its component in $R_k(G)$.

The technical lemma below is our basic tool used to reason about and manipulate reconfiguration sequences.

► **Lemma 5.** *Let $A = (a_1, \dots, a_\ell)$ and $X = (x_1, \dots, x_\ell)$ be two independent sets in G and let $S = ((u_1, v_1), \dots, (u_m, v_m))$ be a reconfiguration sequence from A to X . With $\mathcal{A} = \{S|_t(A) : t \in \{0, \dots, m\}\}$ we denote the set of all independent sets traversed from A to X along S . Suppose that there are $i, j \in \{0, \dots, \ell+1\}$ with $i < j$ such that*

$$\begin{aligned} \text{ex}_i(\mathcal{A}, \text{left}) &= x_i \quad \text{if } i \geq 1, \text{ and} \\ \text{ex}_j(\mathcal{A}, \text{right}) &= x_j \quad \text{if } j \leq \ell. \end{aligned}$$

Then $A' = (x_1, x_2, \dots, x_i, a_{i+1}, \dots, a_{j-1}, x_j, \dots, x_\ell)$ is also an independent set in G . Moreover, if we let S' be S restricted to those pairs (u_t, v_t) with u_t being at a position p with $i < p < j$ in $S|_{t-1}(A)$, then S' is a reconfiguration sequence from A' to X in G .

► **Remark 6.** Since none of the first i tokens nor the last $\ell+1-j$ tokens are affected by S' we can alternatively conclude that S' transforms $(a_{i+1}, \dots, a_{j-1})$ into $(x_{i+1}, \dots, x_{j-1})$ in $G \setminus \bigcup_{p \notin \{i+1, \dots, j-1\}} N[x_p]$

Proof. Let $B = (b_1, \dots, b_\ell)$ be a set in \mathcal{A} . The *aligned* set of B is defined as $(x_1, \dots, x_i, b_{i+1}, \dots, b_{j-1}, x_j, \dots, x_\ell)$. We claim that the aligned set of B is an independent set in G . Note that some of the three parts $Y_1 = (x_1, \dots, x_i)$, $Y_2 = (b_{i+1}, \dots, b_{j-1})$, $Y_3 = (x_j, \dots, x_\ell)$ might be empty. Since all three parts are contained in an independent set, i.e. X or B , all we need to show is that (1) if Y_1 and Y_2 are non-empty, then x_i is completely to the left of b_{i+1} , and (2) if Y_2 and Y_3 are non-empty, then b_{j-1} is completely to the left of x_j . Thus, suppose that Y_1 and Y_2 are non-empty, so x_i and b_{i+1} exist. By the assumptions of the lemma $x_i = \text{ex}_i(\mathcal{A}, \text{left}) \leq_{\text{right}} b_i$ and clearly b_i is completely to the left of b_{i+1} . Therefore, x_i is completely to the left of b_{i+1} as desired. Symmetrically, if Y_2 and Y_3 are non-empty, then $b_{j+1} \leq_{\text{left}} x_{j+1}$ and b_j is completely to the left of b_{j+1} . Therefore, b_j is completely to the left of x_{j+1} as desired.

Since A' is the aligned set of A , we conclude that A' is independent in G . Let m' be the length of S' and $S' = ((u'_1, v'_1), \dots, (u'_{m'}, v'_{m'}))$. Since S' is a subsequence of S , we can fix $\varphi(t)$, for each $t \in \{1, \dots, m'\}$ such that the pair $(u_{\varphi(t)}, v_{\varphi(t)})$ in S corresponds to (u'_t, v'_t) in S' . Let $A_{\varphi(t)} = S|_{\varphi(t)}(A)$ and $A'_t = S'|_t(A')$. By construction we have

$$\begin{aligned} \pi_p(A'_t) &= \pi_p(X) && \text{if } p \in \{1, \dots, i\} \cup \{j, \dots, \ell\} \\ \pi_p(A'_t) &= \pi_p(A_{\varphi(t)}) && \text{if } p \in \{i+1, \dots, j-1\}. \end{aligned}$$

Thus for each $t \in \{0, \dots, m'\}$ we have that $S'|_t(A')$ is the aligned set of $A_{\varphi(t)} \in \mathcal{A}$. Therefore $S'|_t(A')$ is independent in G which completes the proof that S' is a reconfiguration sequence from A' to X . ◀

► **Lemma 7.** *Let H be a non-empty induced subgraph of G , let $\ell \in \{1, \dots, k\}$, and let \mathcal{C} be a component of $R_\ell(H)$. For every $p \in \{0, \dots, \ell\}$, the p -extreme set of \mathcal{C} is independent in H and lies in \mathcal{C} .*

Proof. Fix $p \in \{0, \dots, \ell\}$. Let $X = (x_1, \dots, x_\ell)$ be the p -extreme set of \mathcal{C} . We claim that for every $i \in \{0, \dots, p\}$ and $j \in \{p+1, \dots, \ell+1\}$, there is a set $A_{i,j} \in \mathcal{C}$ such that for all $q \in \{1, \dots, i\} \cup \{j, \dots, \ell\}$, we have $\pi_q(A_{i,j}) = x_q$. We prove this claim by induction on $m = i + (\ell + 1 - j)$. For the base case, when $m = 0$, so $i = 0$ and $j = \ell + 1$, we simply choose $A_{0,\ell+1}$ to be any element in \mathcal{C} .

For the inductive step, consider $m > 0$, so $i > 0$ or $j < \ell + 1$. The two cases are symmetric, so let us consider only the first one. Thus suppose $i > 0$ and therefore by the induction hypothesis, we get an independent set $A_{i-1,j} \in \mathcal{C}$ of the form

$$(x_1, \dots, x_{i-1}, a_i, \dots, a_{j-1}, x_j, \dots, x_\ell),$$

where a_i, \dots, a_{j-1} are some vertices of H . Let $B \in \mathcal{C}$ such that $\pi_i(B) = \text{ex}_i(\mathcal{C}, \text{left}) = x_i$. Since $B, A_{i-1,j} \in \mathcal{C}$, there is a reconfiguration sequence S from B to $A_{i-1,j}$. By Lemma 5, we obtain that

$$(x_1, \dots, x_{i-1}, x_i, b_{i+1}, \dots, b_{j-1}, x_j, \dots, x_\ell) \in \mathcal{C}.$$

This set witnesses the inductive condition for (i, j) and finishes the inductive step. \blacktriangleleft

When $k = 1$, there is a single token in the graph which moves along a path in the interval graph. This is a rather trivial setting still, we give explicit functions (Algorithm 1) to have a good base for the general strategy.

Algorithm 1 A simple algorithm for moving a single token.

```

1: function PUSH_TOKEN_LEFT( $H, u$ )
2:    $w := \min_{\leq_{\text{right}}} \{v \in V(H) : v \text{ is reachable from } u \text{ in } H\}$ 
3:   let  $(v_0, \dots, v_m)$  be the shortest path from  $u$  to  $w$  in  $H$ 
4:   return  $[w, ((v_0, v_1), \dots, (v_{m-1}, v_m))]$ 
5: function PUSH_TOKEN_RIGHT( $H, u$ )
6:    $w := \max_{\leq_{\text{left}}} \{v \in V(H) : v \text{ is reachable from } u \text{ in } H\}$ 
7:   let  $(v_0, \dots, v_m)$  be the shortest path from  $u$  to  $w$  in  $H$ 
8:   return  $[w, ((v_0, v_m), \dots, (v_{m-1}, v_m))]$ 

```

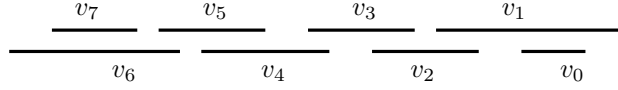
Proposition 8. *Let H be an interval graph, and u be a vertex in H . Then $\text{PUSH_TOKEN_LEFT}(H, u)$ outputs the \leq_{right} -minimum vertex w in the component of u in H and a witnessing u - w path $(v_0, v_1), \dots, (v_{m-1}, v_m)$ in H , where $v_0 = u, v_m = w$ and $v_m <_{\text{right}} \dots <_{\text{right}} v_2 <_{\text{right}} v_1$ and if $m \geq 2$, $v_2 <_{\text{right}} v_0$.*

Symmetrically, $\text{PUSH_TOKEN_RIGHT}(H, u)$ outputs the \leq_{left} -maximum vertex w in the component of u in H and a witnessing u - w path $(v_0, v_1), \dots, (v_{m-1}, v_m)$ in H , where $v_0 = u, v_m = w$ and $v_m >_{\text{left}} \dots >_{\text{left}} v_2 >_{\text{left}} v_1$ and if $m \geq 2$, $v_2 >_{\text{left}} v_0$.

Proof. We prove the first part of the statement about $\text{PUSH_TOKEN_LEFT}(H, u)$. The second part is symmetric. Consider a shortest path $v_0 v_1 \dots v_m$ from u to w in H . Note that there is no point on the line that belongs to three intervals in the path as otherwise we could make the path shorter. It is easy to see that v_{i+1} left overlaps v_i , i.e., $v_{i+1} <_{\text{right}} v_i$ and $v_{i+1} <_{\text{left}} v_i$ for all $i \in \{0, \dots, m-1\}$ possibly except two cases: v_m may be contained in v_{m-1} and (if $m \geq 2$) v_1 may contain v_0 . See Figure 1 for an illustration of such a shortest path in an interval graph. \blacktriangleleft

For $k = 2$, we give an algorithm, see Algorithm 2, that finds a short reconfiguration from a given independent set A in H to the 1-extreme set of the component of A in a reconfiguration graph $R_2(H)$.

23:6 Reconfiguring Independent Sets on Interval Graphs



■ **Figure 1** A shortest path from v_0 to v_7 where v_7 is an interval with leftmost right endpoint.

■ **Algorithm 2** A specialized algorithm finding the 1-extreme set of the component of the given set A in $R_2(H)$.

```

1: function PUSHAPART( $H, A$ )
2:    $a := \pi_1(A)$   $b := \pi_2(A)$ 
3:    $S := ()$ 
4:   do
5:      $[a, S_1] := \text{PUSHTOKENLEFT}(H \setminus N[b], a)$ 
6:      $[b, S_2] := \text{PUSHTOKENRIGHT}(H \setminus N[a], b)$ 
7:      $S := \text{CONCAT}(S, S_1, S_2)$ 
8:   while  $\text{CONCAT}(S_1, S_2) \neq \emptyset$ 
9:   return  $(a, b), S$ 

```

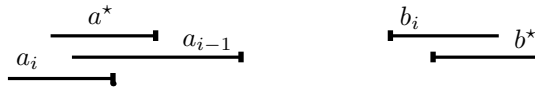
► **Proposition 9.** *Let H be an interval graph and A be an independent set of size 2 in H . Then $\text{PUSHAPART}(H, A)$ outputs the 1-extreme set of the component of A in $R_2(H)$ and a reconfiguration sequence from A to the 1-extreme set of length at most $2|V(H)|$.*

Proof. Suppose that $\text{PUSHAPART}(H, A)$ outputs (a^*, b^*) and let (x_1, x_2) be the 1-extreme set of the component \mathcal{C} of A in $R_2(H)$. Suppose to the contrary that $(a^*, b^*) \neq (x_1, x_2)$. Thus, either $x_1 = \text{ex}_1(\mathcal{C}, \text{left}) <_{\text{right}} a^*$ or $b^* <_{\text{left}} \text{ex}_2(\mathcal{C}, \text{right}) = x_2$.

Consider a path $((a_0, b_0), \dots, (a_m, b_m))$ from (a^*, b^*) to (x_1, x_2) in $R_2(G)$. Let i be the smallest index such that $a_i <_{\text{right}} a_0 = a^*$ or $b^* <_{\text{left}} b_i$. This index is well-defined as $i = m$ satisfies the condition and obviously $i > 0$. Now suppose that $a_i <_{\text{right}} a^*$. The proof of the other case goes symmetrically. By the minimality of i we have

$$a_i <_{\text{right}} a^* \leq_{\text{right}} a_{i-1} \quad \text{and} \quad b_i = b_{i-1} \leq_{\text{left}} b^*.$$

The inequalities on the right endpoints of a_i , a^* , and a_{i-1} imply that these three intervals form a connected subgraph of H , because a_i and a_{i-1} must intersect as they are adjacent. Since a_{i-1} and a_i are in the same component of $H \setminus N[b_i]$, we conclude that a^* is also in that component. Finally, since $b_i \leq_{\text{left}} b^*$ we conclude that a_{i-1} , a^* , a_i are together in the same component of $H \setminus N[b^*]$. See Figure 2 that illustrates all these inequalities. Consider the last



■ **Figure 2** $a_i <_{\text{right}} a^* \leq_{\text{right}} a_{i-1}$ and $b_i \leq_{\text{left}} b^*$. The intervals a_i , a^* , a_{i-1} are together in a component of $H \setminus N[b_i]$ and so they are together in a component of $H \setminus N[b^*]$ as well.

iteration of the loop in $\text{PUSHAPART}(H, A)$. The variables a and b keep the values a^* and b^* in this iteration. In particular $\text{PUSHTOKENLEFT}(H \setminus N[b^*], a^*)$ did not change the value of a . This is a contradiction as by Proposition 8 $\text{PUSHTOKENLEFT}(H \setminus N[b^*], a^*)$ outputs the \leq_{right} minimum vertex in the component of a^* in $H \setminus N[b^*]$ but we already know that a^* is not \leq_{right} -minimum there.

Now, let us prove the claim about the length of the output reconfiguration sequence S . Let $(a_0, b_0), \dots, (a_m, b_m)$ be the path in $R_2(H)$ corresponding to S , that is $(a_t, b_t) = S|_t(A)$ for $t \in \{0, \dots, m\}$. Suppose that $(a, b) = (a^0, b^0)$ at the beginning of the loop in line 4 and $(a, b) = (a^1, b^1)$ after pushing left and right in lines 5-6. By Proposition 8, we know that $a^1 \leq_{\text{right}} a^0$ and $b^0 \leq_{\text{left}} b^1$ and every intermediate configuration (a', b') on the path satisfies $a^1 \leq_{\text{right}} a' \leq_{\text{right}} a^0$ and $b^0 \leq_{\text{left}} b' \leq_{\text{left}} b^1$ with a possible exception of the first move from a^0 and the first move from b^0 . This way we see that the total number of steps without the exceptional moves is at most $|V(H)|$ and each interval in H can be the target of at most one exceptional move. This gives a bound $2|V(H)|$, as desired. \blacktriangleleft

We present now our main reconfiguration algorithm, see Algorithm 3.

Algorithm 3 A general algorithm finding the $(k-1)$ -extreme set of the component of the given set A in $R_k(G)$.

```

1: function RECONFIGURE( $G, k, A$ )
2:   for  $i \in \{1, 2, \dots, k\}$  do
3:      $a_i := \pi_i[A]$ 
4:      $\text{lex}_i := a_i$   $\triangleright \text{lex}_k$  is unused
5:      $\text{rex}_i := a_i$   $\triangleright \text{rex}_1$  is unused
6:    $j := 1$ ;  $S = ()$ 
7:   while  $j < k$  do
8:      $[(a_j, a_{j+1}), S'] := \text{PUSHAPART}(G \setminus \bigcup_{i \neq j, j+1} N[a_i], \{a_j, a_{j+1}\})$ 
9:      $S.\text{APPEND}(S')$ 
10:     $\gamma = 1$ 
11:    if  $(a_j, a_{j+1}) \neq (\text{lex}_j, \text{rex}_{j+1})$  then
12:       $\text{lex}_j := a_j$ 
13:       $\text{rex}_{j+1} := a_{j+1}$ 
14:      if  $j > 1$  then  $\gamma = -1$ 
15:     $j := j + \gamma$ 
16:  return  $(a_1, \dots, a_k), S$ 

```

► Lemma 10. *Let $k \geq 2$ and let A be an independent set of size k in G . Then $\text{RECONFIGURE}(G, k, A)$ outputs the $(k-1)$ -extreme set of the component of A in $R_k(G)$ and a reconfiguration sequence from A to this set of length $\mathcal{O}(k \cdot n^2)$.*

Proof. We begin our consideration of Algorithm 3 by noting two invariants.

\triangleright **Claim 11.** Every time Algorithm 3 reaches line 15, we have

$$(a_1, \dots, a_k) = (\text{lex}_1, \dots, \text{lex}_j, \text{rex}_{j+1}, \dots, \text{rex}_k).$$

Proof. Note that the equation holds after the initialization in lines 4-5. Later on, the values of (a_1, \dots, a_k) are updated only in line 8 and if so then corresponding values of lex and rex are updated in lines 12-13. \blacktriangleleft

\triangleright **Claim 12.** For every $j \in \{2, \dots, k\}$, the values held by lex_j are nonincreasing with respect to \leq_{right} . Symmetrically, for every $j \in \{1, \dots, k-1\}$, the values held by rex_j are nondecreasing with respect to \leq_{left} .

Proof. We prove the statement by induction over the iterations steps in Algorithm 3. Consider the moment when the values of lext_j and rext_{j+1} are updated in lines 12-13. Let $(\ell_1, \ell_2, \dots, \ell_k)$ and (r_1, r_2, \dots, r_k) be the values held by vectors lext and rext just after this update. Let $(\ell'_1, \dots, \ell'_k)$ and (r'_1, \dots, r'_k) be the values held by vectors lext and rext just after the previous update of lext_j or rext_{j+1} in lines 12-13 or the initial values of these vectors (assigned in lines 4-5) if there was no previous update. All we need to show is that $\ell_j \leq_{\text{right}} \ell'_j$ and $r'_{j+1} \leq_{\text{left}} r_{j+1}$.

By Claim 11 we know that the algorithm had tokens in $(\ell'_1, \dots, \ell'_j, r'_{j+1}, \dots, r'_k)$ and after applying some reconfiguration sequence, say $S = ((u_1, v_1), \dots, (u_m, v_m))$, it reached the configuration $(\ell_1, \dots, \ell_j, r_{j+1}, \dots, r_k)$. Let $\mathcal{A} = \{S|_t(A) : t \in \{0, \dots, m\}\}$.

By induction hypothesis the statement holds for all the updates applied so far by Algorithm 3. In particular, ℓ_{j-1} is the \leq_{right} -minimal position of the $(j-1)$ -th token so far and r_{j+2} is the \leq_{left} -maximal position of the $(j+2)$ -th token so far (assuming that these tokens exist). Thus,

$$\begin{aligned} \ell_{j-1} &= \text{ex}_{j-1}(\mathcal{A}, \text{left}) && \text{if } j-1 \geq 1, \\ r_{j+2} &= \text{ex}_{j+2}(\mathcal{A}, \text{right}) && \text{if } j+2 \leq k. \end{aligned}$$

Therefore, we may apply Lemma 5 and Remark 6 and conclude that (ℓ'_j, r'_{j+1}) and (ℓ_j, r_{j+1}) are in the same component of $R_2(G \setminus \bigcup_{i \neq j, j+1} N[a_i])$. By Proposition 9, the execution of

PUSHAPART in line 8 outputs the 1-extreme set, namely (ℓ_j, r_{j+1}) , of the component of (ℓ'_j, r'_{j+1}) in $R_2(G \setminus \bigcup_{i \neq j, j+1} N[a_i])$. Therefore, $\ell_j \leq_{\text{right}} \ell'_j$ and $r_{j+1} \geq_{\text{left}} r'_{j+1}$, as desired. \triangleleft

\triangleright Claim 13. Consider a moment when Algorithm 3 reaches the line 15 and $\gamma = 1$. Let α be the value of variable j prior to the update. Let (ℓ_1, \dots, ℓ_k) , (r_1, \dots, r_k) be the values held at this moment by vectors lext and rext , respectively. Then,

$$\begin{aligned} \ell_i &= \text{ex}_i(\mathcal{C}, \text{left}) && \text{for } i \in \{1, \dots, \alpha\}, \\ r_{\alpha+1} &= \text{ex}_{\alpha+1}(\mathcal{C}, \text{right}), \end{aligned}$$

where \mathcal{C} is the component of $(\ell_1, \dots, \ell_\alpha, r_{\alpha+1})$ in $R_{\alpha+1}(G \setminus \bigcup_{i > \alpha+1} N[r_i])$. In particular, $(\ell_1, \dots, \ell_\alpha, r_{\alpha+1})$ is the α -extreme set in \mathcal{C} .

Proof. We proceed by induction on α . First we deal with $\alpha = 1$. When Algorithm 3 starts an iteration of the while loop with $j = 1$, then by Claim 11 (and initialization in lines 4-5), we have $(a_3, \dots, a_k) = (r_3, \dots, r_k)$. By Proposition 9, PUSHAPART($G \setminus \bigcup_{i > 2} N[r_i], \{a_1, a_2\}$) executed in line 8 outputs the 1-extreme set of the component of $\{a_1, a_2\}$ in $R_2(G \setminus \bigcup_{i > 2} N[r_i])$. After the update in lines 12-13, this set is stored in $\{\text{lext}_1, \text{rext}_2\} = \{\ell_1, r_2\}$ when Algorithm 3 reaches the line 15, as desired.

Let us assume that $\alpha > 1$ and that the claim holds for all smaller values of α . Consider an iteration of the while loop with $j = \alpha$ such that Algorithm 3 reaches line 15 with $\gamma = 1$. Let (ℓ_1, \dots, ℓ_k) , (r_1, \dots, r_k) be the values held at this moment by vectors lext and rext , respectively. For convenience, we call this iteration the *present* iteration.

Now starting from the present iteration consider the last iteration before with $j = \alpha - 1$. We call this iteration the *past* iteration. Clearly, the past iteration had to conclude with $\gamma = 1$ and all iterations between the past and the present (there could be none) must have the value of variable $j \geq \alpha$ and those with $j = \alpha$ must conclude with $\gamma = 1$. This implies that the values of $(\text{lext}_1, \dots, \text{lext}_\alpha)$ and $(\text{rext}_2, \dots, \text{rext}_{\alpha+1})$ did not change between the past and the present iterations so they constantly are $(\ell_1, \dots, \ell_\alpha)$ and $(r_2, \dots, r_{\alpha+1})$.

Let \mathcal{D} be the connected component of $(\ell_1, \ell_2, \dots, \ell_{\alpha-1}, r_\alpha)$ in $R_\alpha(G \setminus N[r_{\alpha+1}])$. The inductive assumption for the past iteration yields:

$$\begin{aligned} \ell_i &= \text{ex}_i(\mathcal{D}, \text{left}) && \text{for } i \in \{1, \dots, \alpha-1\}, \text{ and} && (\star) \\ r_\alpha &= \text{ex}_\alpha(\mathcal{D}, \text{right}), \end{aligned}$$

Note that in the iteration immediately following the past iteration (this was an iteration with $\alpha = j$ and may be the present iteration) the only token movement was the travel of the j -th token from r_α to ℓ_α (the $(j+1)$ -th token stays at $r_{\alpha+1}$). In particular, there is a path from r_α to ℓ_α in $G \setminus (N[\ell_{\alpha-1}] \cup N[r_{\alpha+1}])$. Therefore, there is a path connecting $(\ell_1, \dots, \ell_{\alpha-1}, r_\alpha)$ and $(\ell_1, \dots, \ell_\alpha)$ in $R_\alpha(G \setminus N[r_{\alpha+1}])$, so both independent sets are in \mathcal{D} .

Now we argue, that

$$\ell_\alpha = \text{ex}_\alpha(\mathcal{D}, \text{left}).$$

Indeed, take any $(v_1, v_2, \dots, v_\alpha) \in \mathcal{D}$ and we aim to show that $\ell_\alpha \leq_{\text{right}} v_\alpha$. As (v_1, \dots, v_α) and $(\ell_1, \dots, \ell_\alpha)$ are in one component of the reconfiguration graph $R_\alpha(G \setminus N[r_{\alpha+1}])$ and by (\star) , we may apply Lemma 5 to conclude that $(\ell_1, \ell_2, \dots, \ell_{\alpha-1}, v_\alpha)$ also lies in \mathcal{D} . Moreover by Remark 6, the vertices ℓ_α and v_α are connected by a path in $G \setminus (N[\ell_{\alpha-1}] \cup N[r_{\alpha+1}])$. Thus by Proposition 9, PUSHAPART executed in the present iteration guarantees that $\ell_\alpha \leq_{\text{right}} v_\alpha$ as claimed.

Let \mathcal{C} be the component of $(\ell_1, \dots, \ell_\alpha, r_{\alpha+1})$ in $R_{\alpha+1}(G \setminus \bigcup_{i>\alpha+1} N[r_i])$. We proceed to argue that

$$r_{\alpha+1} = \text{ex}_{\alpha+1}(\mathcal{C}, \text{right}).$$

Assume for the sake of contradiction that there is some $A \in \mathcal{C}$, such that $r_{\alpha+1} <_{\text{left}} \pi_{\alpha+1}(A)$ and fix such an A with a shortest possible reconfiguration sequence $S = ((u_1, v_1), \dots, (u_m, v_m))$ from $(\ell_1, \dots, \ell_\alpha, r_{\alpha+1})$ to A in $R_{\alpha+1}(G \setminus \bigcup_{i>\alpha+1} N[r_i])$. Let $A_t = S|_t((\ell_1, \dots, \ell_\alpha, r_{\alpha+1}))$, for all $t \in \{0, \dots, m\}$. By the choice of A and S , for every $t \in \{0, \dots, m-1\}$ we have $\pi_{\alpha+1}(A_t) \leq_{\text{left}} r_{\alpha+1}$. We apply now Lemma 5 (with $i = 0$, $j = \alpha + 1$) to a path from $A_0 = (\ell_1, \dots, \ell_\alpha, r_{\alpha+1})$ to A_{m-1} and conclude that for each $t \in \{0, \dots, m-1\}$ the set $A'_t = (\pi_1(A_t), \dots, \pi_\alpha(A_t), r_{\alpha+1})$ is an independent set in \mathcal{C} . Consider now a path of independent sets of size α formed by dropping the $(\alpha + 1)$ -th coordinate of each set in the path $(A'_0, \dots, A'_{m-1}, A_m)$. Since $(\pi_1(A_0), \dots, \pi_\alpha(A_0)) = (\ell_1, \dots, \ell_\alpha) \in \mathcal{D}$, the whole path lives in \mathcal{D} . Therefore, by (\star) we have

$$\ell_i \leq_{\text{right}} \pi_i(A_t),$$

for all $t \in \{0, \dots, m\}$ and $i \in \{1, \dots, \alpha\}$. But this in turn allows us to apply Lemma 5 and Remark 6 once more (this time with $i = \alpha$, $j = \alpha + 2$) to a path from $(\ell_1, \dots, \ell_\alpha, r_{\alpha+1})$ to A_m and we conclude that $r_{\alpha+1}$ and $\pi_{\alpha+1}(A_m)$ are in the same component of $G \setminus \left(N[\ell_\alpha] \cup \bigcup_{i>\alpha+1} N[r_i] \right)$. But PUSHAPART executed in the present iteration outputs $(\ell_\alpha, r_{\alpha+1})$ while $r_{\alpha+1} <_{\text{left}} \pi_{\alpha+1}(A_m)$. This contradicts Proposition 9 and completes the proof that $r_{\alpha+1} = \text{ex}_{\alpha+1}(\mathcal{C}, \text{right})$.

It remains to prove that $\ell_i = \text{ex}_i(\mathcal{C}, \text{left})$ for all $i \in \{1, \dots, \alpha\}$. Pick an arbitrary $A = (v_1, v_2, \dots, v_{\alpha+1}) \in \mathcal{C}$. Since we already know that $r_{\alpha+1} = \text{ex}_{\alpha+1}(\mathcal{C}, \text{right})$, we can apply Lemma 5 (with $i = 0$, $j = \alpha + 1$) to a path from $(\ell_1, \dots, \ell_\alpha, r_{\alpha+1})$ to $(v_1, v_2, \dots, v_{\alpha+1})$, and we conclude that there is a reconfiguration sequence transforming (v_1, \dots, v_α) into $(\ell_1, \dots, \ell_\alpha)$ in $G \setminus N[r_{\alpha+1}]$. Thus, this path lies in \mathcal{D} and the desired inequalities $\ell_i \leq_{\text{right}} v_i$ for all $i \in \{1, \dots, \alpha\}$ follow by (\star) . \triangleleft

23:10 Reconfiguring Independent Sets on Interval Graphs

Clearly, Claims 13 and 11 establish the correctness of the algorithm. Equipped with the invariant given by Claim 12, we can bound the length of the returned reconfiguration sequence. Indeed, observe that in each iteration of the while loop in line 7 either lext_j decreases wrt. \leq_{right} , or rext_{j+1} increases wrt. \leq_{left} while j drops by 1, or j increases by 1. Now this implies that the outer loops can iterate at most $4nk + k$, as the quantity

$$j + 2 \sum_{i=1}^k \text{Index}_{\leq_{\text{left}}}(\text{rext}_i) + (n - \text{Index}_{\leq_{\text{right}}}(\text{lext}_i) + 1),$$

where $\text{Index}_{\leq}(x)$ denotes the position of element x in a given linear order \leq on some fixed finite set, increases by at least one in each iteration and it is at most $4nk + k$.

As seen in Proposition 9, each call of the procedure PUSHAPART returns a sequence consisting of at most $2n$ moves. Therefore, the length of reconfiguration sequence returned by Algorithm 3 is at most $8kn^2 + 2kn \in \mathcal{O}(kn^2)$. This completes the proof of Theorem 1. ◀

4 Lower bound: The Example

We present a family of graphs $\{G_{m,k}\}_{m,k \geq 1}$, such that $|V(G_{m,k})| = 8k + 2m - 5$ and $R_k(G_{m,k})$ contains a component of diameter at least $\frac{k^2}{4} \cdot m$. This will prove Theorem 2.

Fix integers $m, k \geq 1$. We will describe a family of intervals $\mathcal{I}_{m,k}$. The graph $G_{m,k}$ will be simply the intersection graph of $\mathcal{I}_{m,k}$. We construct the family in three steps. We initialize $\mathcal{I}_{m,k}$ with $(k-1) + (m+2k-1) + k$ pairwise disjoint intervals:

$$a_{k-1}, \dots, a_1, v_1, \dots, v_{m+2k-1}, b_1, \dots, b_k,$$

listed with their natural left to right order on the line. We call these intervals, the *base* intervals. Let $N = m + 2k - 1$. We put into $\mathcal{I}_{m,k}$ further $N - 1$ intervals:

$$v_{1,2}, v_{2,3}, \dots, v_{N-1,N},$$

where for each $i \in \{1, \dots, N-1\}$, the interval $v_{i,i+1}$ is an open interval with the left endpoint in the middle of v_i and the right endpoint in the middle of v_{i+1} . We call these intervals the *path* intervals. Finally, we put into $\mathcal{I}_{m,k}$ two groups of *long* intervals:

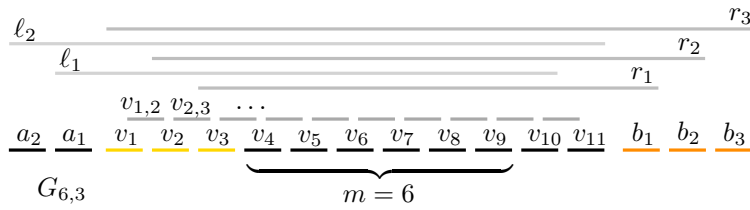
$$\ell_1, \dots, \ell_{k-1} \text{ and } r_1, \dots, r_k,$$

where for each $i \in \{1, \dots, k-1\}$ the interval ℓ_i is the open interval with the left endpoint coinciding with the left endpoint of a_i and the right endpoint coinciding with the right endpoint of $v_{N-(k-1)-i}$. Symmetrically, for each $i \in \{1, \dots, k\}$ the interval r_i is the open interval with the left endpoint coinciding with the left endpoint of v_{k-i+1} and the right endpoint coinciding with the right endpoint of b_i . This completes the construction of $\mathcal{I}_{m,k}$. See Figure 3.

Consider two independent sets $I = (v_1, \dots, v_k)$ and $J = (b_1, \dots, b_k)$ in $G_{m,k}$.

► **Lemma 14.** *The sets I and J are in the same component of $R_k(G_{m,k})$ and every reconfiguration sequence from I to J has length at least $\frac{k^2}{4} \cdot m$.*

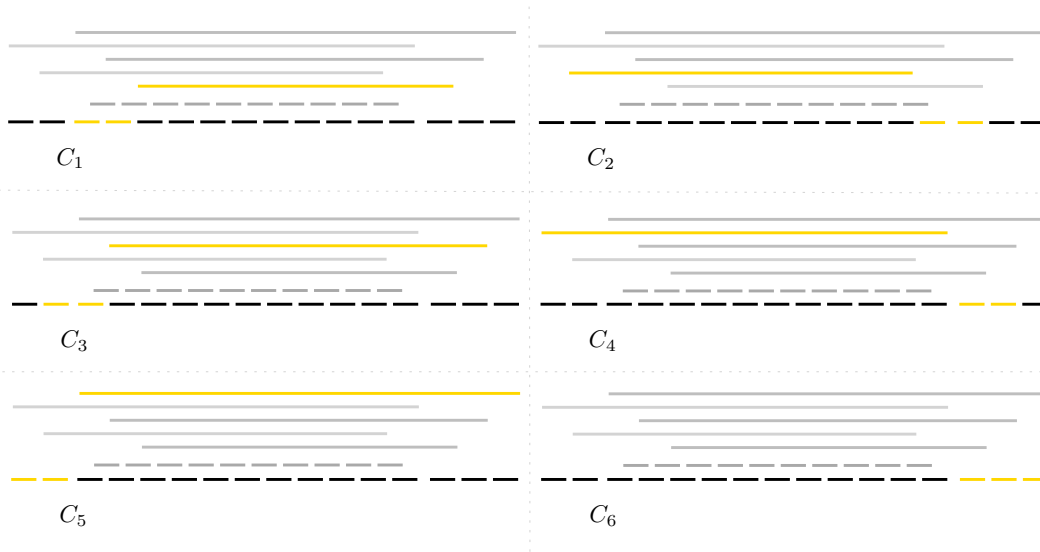
Proof. We put most of the effort to prove the second part of the statement, that every reconfiguration sequence from I to J has length at least $\frac{k^2}{4} \cdot m$.



■ **Figure 3** The graph $G_{6,3}$ with two distinguished independent sets $I = \{v_1, v_2, v_3\}$ and $J = \{b_1, b_2, b_3\}$.

We define a sequence of independent sets (see Figure 4):

$$\begin{aligned}
 C_0 &= (v_1, \dots, v_k) = I, \\
 C_1 &= (v_1, \dots, v_{k-1}, r_1), \\
 C_2 &= (\ell_1, v_{N-(k-1)}, \dots, v_N, b_1), \\
 &\vdots \\
 C_{2i-1} &= (a_{i-1}, \dots, a_1, v_1, \dots, v_{k-1}, r_i), \\
 C_{2i} &= (\ell_i, v_{N-(k-i-2)}, \dots, v_N, b_1, \dots, b_i), \\
 &\vdots \\
 C_{2k} &= (b_1, \dots, b_k) = J.
 \end{aligned}$$



■ **Figure 4** The sets C_1, \dots, C_6 in $G_{6,3}$.

It is easy to construct a path from C_j to C_{j+1} in $R_k(G_{m,k})$ for $j \in \{0, \dots, 2k - 1\}$ which proves that I, J are in the same component of $R_k(G_{m,k})$.

Let (K_0, \dots, K_M) be a path in $R_k(G_{m,k})$ from I to J . The proof will follow from two claims. The first one is that (C_0, \dots, C_{2k}) is a subsequence of (K_0, \dots, K_M) , and the second one is that for every $i \in \{1, \dots, k - 1\}$ every path from C_{2i-1} to C_{2i} is of length at least $(k - 2i - 1) \cdot m$. A symmetric argument can be used to bound the distance between C_{2i} and C_{2i+1} which we omit here as it would only improve the final lower bound by a constant factor.

23:12 Reconfiguring Independent Sets on Interval Graphs

Let P be the set of path intervals in $G_{m,k}$. Define for each $i \in \{1, \dots, k-1\}$ the following graphs:

$$H_{2i-1} = G_{m,k}[\{a_{i-1}, \dots, a_1, v_1, \dots, v_N, b_1, \dots, b_i\} \cup P],$$

$$H_{2i} = G_{m,k}[\{a_i, \dots, a_1, v_1, \dots, v_N, b_1, \dots, b_i\} \cup P].$$

Note that $C_0 = K_0$ and $C_{2k} = K_M$, so C_0 and C_{2k} occurs in (K_0, \dots, K_M) . Fix $j \in \{0, \dots, 2k-2\}$. Suppose that the independent set C_j occurs in (K_0, \dots, K_M) and fix such an occurrence. We will argue that C_{j+1} must occur afterwards in the sequence.

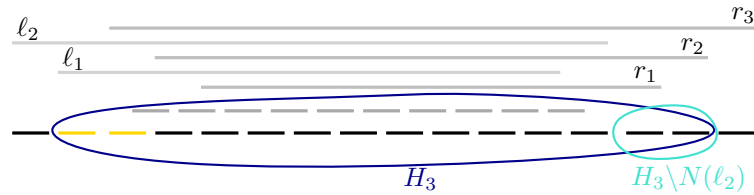
Observe that all base intervals from C_j are in H_j . However $b_k \notin H_j$ and $b_k \in K_M$, hence to reconfigure from C_j to K_M eventually a token has to be moved to some base interval not in H_j . Thus, let X_j be the set of base intervals not in H_j , i.e.

$$X_j = \begin{cases} \{a_{k-1}, \dots, a_i\} \cup \{b_{i+1}, \dots, b_k\} & \text{if } j \text{ is odd,} \\ \{a_{k-1}, \dots, a_{i+1}\} \cup \{b_{i+1}, \dots, b_k\} & \text{if } j \text{ is even.} \end{cases}$$

Note that the only neighbours of intervals in X_j are long. Let Y be the first independent set in (K_0, \dots, K_M) that occurs after the fixed occurrence of C_j and contains a long interval u_0 neighbouring some element in X_j . We claim that $Y = C_{j+1}$.

First, we show that $u_0 = \ell_i$ if $j = 2i-1$ and $u_0 = r_{i+1}$ if $j = 2i$ respectively. Assume for now that $j = 2i-1$. Observe that for all $p \in \{1, \dots, i-1\}$ we have $N(\ell_p) \cap X_j = \emptyset$ and consequently $u_0 \neq \ell_p$. On the other hand, for all $p \in \{i+1, \dots, k-1\}$ we have $\alpha(H_j \setminus N(\ell_p)) < k-1$. Therefore, whenever $u_0 = \ell_p$ there is a token in Y that is not in H_j . This contradicts the minimality of Y . Moreover, for all $p \in \{1, \dots, i+1\}$ we have $N(r_p) \cap X_j = \emptyset$ in turn implying that $u_0 \neq r_p$. On the other hand, for all $p \in \{i+2, \dots, k-1\}$ we have $\alpha(H_j \setminus N(r_p)) < k-1$, thus, $u_0 \neq r_p$. This leaves only one possible option of $u_0 = \ell_i$. The case $j = 2i$ follows a symmetric argument. See Figure 5.

Recall that all $k-1$ elements of $Y \setminus \{u_0\}$ must be in H_j . It is easy to see that when $j = 2i-1$ then $H_j \setminus N(\ell_{i+1})$ has exactly one independent set of size $k-1$, namely: $\{v_{N-(k-i-2)}, \dots, v_N, b_1, \dots, b_i\}$, symmetrically when $j = 2i$ then $H_j \setminus N(r_{i+1})$ has exactly one independent set of size $k-1$, namely: $\{a_i, \dots, a_1, v_1, \dots, v_{k-1}\}$. This proves that $Y = C_{j+1}$.



■ **Figure 5** The set H_3 for $G_{6,3}$. We interpret $H_3 \setminus N(u)$ as the space where $k-1$ tokens can “hide”. All base neighbours of ℓ_1, r_1 , and r_2 are in H_3 . Also, $\alpha(H_3 \setminus N(r_3)) = 1 < 2$. This gives $u_0 = \ell_2$.

Let us now prove that for a fixed $i \in \{1, \dots, k-1\}$ every path from C_{2i-1} to C_{2i} in $R_k(G_{m,k})$ is of length at least $(k-2i-1) \cdot m$.

Fix the shortest reconfiguration sequence from C_{2i-1} to C_{2i} . As tokens do not interchange their relative positions, note that tokens starting at the positions (v_1, \dots, v_{k-2i}) in C_{2i-1} must finish at the positions $(v_{N-(k-2i-1)}, \dots, v_N)$ in C_{2i} . We call these tokens *heavy*. Their left to right ordinal numbers are $i, \dots, k-i$, and there are exactly $s := k-2i-1$ of them.

We prove that a heavy token cannot use any of the long intervals during the reconfiguration. By the first part of the proof we know that on the shortest path from C_{2i-1} to C_{2i} only base intervals from H_{2i-1} can be used. For each long interval u we define H_u^ℓ as a graph induced by all intervals v in $H_{2i-1} \setminus N(u)$ completely to the left of u . Analogously, define H_u^r as the graph induced by all $v \in H_{2i-1} \setminus N(u)$ completely to the right to u . Finally, put

$$n_\ell(u) = \alpha(H_u^\ell) \text{ and } n_r(u) = \alpha(H_u^r).$$

Assume that a heavy token uses a fixed long interval w on the path from C_{2i-1} to C_{2i} . Armed with the knowledge of the ordinal numbers of the heavy tokens, we see that: $n_\ell(w) \geq i-1$ and $n_r(w) \geq i$. Elementary computation shows that for every long interval u either $n_\ell(u) < i-1$ or $n_r(u) < i$, which proves that no such long interval w exists.

As heavy tokens cannot use long intervals, each of them has to use base and path intervals forcing it to make at least $2(N-s+1) \geq m$ steps. Therefore, we need at least $s \cdot m$ steps in the path.

Summing up all required steps, we conclude, that every path from I to J in $R_k(G_{m,k})$ has length at least $\frac{k^2}{4} \cdot m$. ◀

5 Hardness result for incomparability graphs

In this section, we present a simple reduction showing that Independent Set Reconfiguration is PSPACE-hard on incomparability graphs in general. Note that interval graphs are incomparability graphs of interval orders. The proof exhibits a reduction from H -Word Reachability defined in [10]. For the readers' convenience we state the definition of this problem here. If H is a digraph (possibly with loops) and $a = a_1 a_2 \dots a_n \in V(H)^*$ then a is an H -word, if for any $i \in \{1, \dots, n-1\}$ we have $a_i a_{i+1} \in E(H)$. In the H -Word Reachability we are given two H -words of the same length a and b , and the question is whether one can transform a into b by changing one letter at a time in such a way that each intermediate word is an H -word.

► **Theorem 15** ([10], Theorem 3). *There exists a digraph H for which the H -Word Reachability is PSPACE-complete.*

► **Theorem 16.** *There exists a constant $w \in \mathbb{N}$, such that Independent Set Reconfiguration is PSPACE-hard on incomparability graphs of posets of width at most w .*

Proof. We demonstrate a reduction from H -Word Reachability for arbitrary H ; the result will follow from Theorem 15.

Fix an instance of H -Word Reachability consisting of two H -words a and b of equal length n . We will construct a poset of width at most $2|V(H)|$, and two independent sets A, B in its incomparability graph, such that A is reconfigurable to B if and only if our starting instance is a yes instance of H -Word Reachability. Define the poset $P_n(H)$ as $(V(H) \times \{1, \dots, n\}, \prec)$ where \prec is defined as follows:

$$(x, i) \prec (y, j) \iff (j = i + 1 \text{ and } xy \in E(H)) \text{ or } (j > i + 1).$$

By the definition of \prec each set of the form $V(H) \times \{i\}$ is an antichain, thus for any chain C in $P_n(H)$ of cardinality n and any $i \in \{1, \dots, n\}$, we have $|(V(H) \times \{i\}) \cap C| = 1$. Therefore, any chain C of cardinality n , can be written as $C = \{(x_1, 1), (x_2, 2), \dots, (x_n, n)\}$. Observe that for each $i \in \{1, \dots, n-1\}$ we have $(x_i, i) \prec (x_{i+1}, i+1) \iff x_i x_{i+1} \in E(H)$. This implies that the first coordinates $x_1 x_2 \dots x_n$ of the elements of chain C form an H -word. Conversely,

given an H -word consisting of n letters $y_1y_2\dots y_n$ the set $\{(y_1, 1), (y_2, 2), \dots, (y_n, n)\}$ is a chain of cardinality n in $P_n(H)$. It follows that a word $x_1x_2\dots x_n$ is an H -word if and only if $\{(x_1, 1), \dots, (x_n, n)\}$ is an independent set in the incomparability graph $\text{Inc}(P_n(H))$.

Let $a = a_1a_2\dots a_n$ and $b = b_1b_2\dots b_n$ be the two given H -words of length n . We define $A = \{(a_1, 1), (a_2, 2), \dots, (a_n, n)\}$ and $B = \{(b_1, 1), (b_2, 2), \dots, (b_n, n)\}$. These are two independent sets in $\text{Inc}(P_n)$. Using the fact that for each $i \in \{1, \dots, n\}$ the set $V(H) \times \{i\}$ is a clique in $\text{Inc}(P_n(H))$, we infer that each edge in $R_n(P_n)$ corresponds to a move of the form $((x, i), (y, i))$ for some $i \in \{1, \dots, n\}$. Thus A is reconfigurable into B if and only if one can transform a into b one letter at a time keeping each intermediate word an H -word.

All that remains is to observe that we can construct the incomparability graph of $P_n(H)$ together with the sets A and B for a fixed H in logarithmic space, and that the width of $P_n(H)$ is always at most $2|V(H)|$. ◀

References

- 1 Marthe Bonamy and Nicolas Bousquet. Token sliding on chordal graphs. In *Graph-Theoretic Concepts in Computer Science, Proc. WG 2017*, volume 10520 of *LNCS*, pages 127–139. Springer, 2017.
- 2 Paul Bonsma, Marcin Kamiński, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In R. Ravi and Inge Li Gørtz, editors, *Algorithm Theory, Proc. SWAT 2014*, volume 8503 of *LNCS*, pages 86–97. Springer, 2014.
- 3 Nicolas Bousquet. Talk at the graph theory seminar in Bordeaux, October 02, 2020. URL: <https://webconf.u-bordeaux.fr/b/mar-ef4-zed>.
- 4 Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Polynomial-time algorithm for sliding tokens on trees. In Hee-Kap Ahn and Chan-Su Shin, editors, *Algorithms and Computation, Proc. ISAAC 2014*, volume 8889 of *LNCS*, pages 389–400. Springer, 2014.
- 5 Eli Fox-Epstein, Duc A Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In *Algorithms and Computation, Proc. ISAAC 2015*, volume 9472 of *LNCS*, pages 237–247. Springer, 2015.
- 6 Robert A. Hearn and Erik D. Demaine. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1):72–96, 2005.
- 7 Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012.
- 8 Daniel Lokshtanov and Amer E. Mouawad. The complexity of independent set reconfiguration on bipartite graphs. *ACM Trans. Algorithms*, 15(1):19 pages, 2018. Article No.: 7.
- 9 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.
- 10 Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *Journal of Computer and System Sciences*, 93:1–10, 2018.

Finite Convergence of μ -Calculus Fixpoints on Genuinely Infinite Structures

Florian Bruse

School of Electrical Engineering and Computer Science, University of Kassel, Germany

Marco Sälzer

School of Electrical Engineering and Computer Science, University of Kassel, Germany

Martin Lange

School of Electrical Engineering and Computer Science, University of Kassel, Germany

Abstract

The modal μ -calculus can only express bisimulation-invariant properties. It is a simple consequence of Kleene's Fixpoint Theorem that on structures with finite bisimulation quotients, the fixpoint iteration of any formula converges after finitely many steps. We show that the converse does not hold: we construct a word with an infinite bisimulation quotient that is locally regular so that the iteration for any fixpoint formula of the modal μ -calculus on it converges after finitely many steps. This entails decidability of μ -calculus model-checking over this word. We also show that the reason for the discrepancy between infinite bisimulation quotients and trans-finite fixpoint convergence lies in the fact that the μ -calculus can only express regular properties.

2012 ACM Subject Classification Theory of computation \rightarrow Modal and temporal logics; Theory of computation \rightarrow Higher order logic; Theory of computation \rightarrow Logic and verification; Theory of computation \rightarrow Automata over infinite objects

Keywords and phrases temporal logic, fixpoint iteration, bisimulation

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.24

1 Introduction

The modal μ -calculus \mathcal{L}_μ , as it was introduced by Kozen [17], has become a de-facto standard yardstick amongst formal specification languages for programs. It is obtained in a principally simple way, namely by extending standard modal logic with extremal fixpoint quantifiers. Since most operators used in temporal logics can be characterised as least or greatest fixpoints, \mathcal{L}_μ can embed standard temporal logics like CTL, LTL and CTL* [10]. \mathcal{L}_μ is also, in a sense, the largest regular program specification logic as it is equi-expressive to the bisimulation-invariant fragment of Monadic Second-Order Logic [13]. Hence, studying its model-theoretic properties helps to answer questions after what can and cannot be formally expressed about programs in regular specification languages.

Fixpoint formulas in \mathcal{L}_μ denote sets of states in a labelled transition system (LTS), and Kleene's Fixpoint Theorem [16] can be used to approximate such fixpoints in a chain of sets: for instance, the semantics of some fixpoint definition $\mu X.\varphi(X)$ can be approximated from below by the sequence of sets X^i , where $X^0 = \emptyset$ and X^{i+1} is obtained as the semantics of $\varphi(X^i)$. For infinite structures, it is generally necessary to extend this sequence to trans-finite ordinal numbers. Over any LTS whose state space forms a set, this sequence must stabilise eventually at precisely the least fixpoint of φ .

Recent times have seen increased interest in the details of this process, regarding questions of when exactly it stabilises, and how this depends on the underlying structure and formula in question. Such questions are not simplified by fixpoint alternation – the ability to nest mutually dependent fixpoint definitions of different kinds. It is known that, over the class



© Florian Bruse, Marco Sälzer, and Martin Lange;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 24; pp. 24:1–24:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of all LTS, this is unavoidable in that alternation-free formulas do not capture all of \mathcal{L}_μ 's expressiveness [5]. However, some classes of structures are known, for instance words, over which any \mathcal{L}_μ formula is equivalent to an alternation-free one [15].

The first notable result regarding the question of fixpoint stabilisation in \mathcal{L}_μ is that it is decidable whether an \mathcal{L}_μ formula is equivalent to a modal one [20]. Other and more recent research concerns the ordinals at which fixpoint iteration stabilises, in particular which ordinals are candidates for such a bound. This concerns the question whether, for a given formula $\mu X.\varphi$, there is some ordinal α such that fixpoint iteration X^0, X^1, \dots stabilises after at most α steps over *any* LTS. Such a (minimal) α is called the *closure ordinal* of $\mu X.\varphi$. Czarnecki [9] shows that, for each ordinal $\alpha < \omega^2$, there is some \mathcal{L}_μ formula with closure ordinal α . Afshari and Leigh [1] show that, for alternation-free formulas, ω^2 is a tight upper bound for closure ordinal candidates. [12] shows that ω_1 is the closure ordinal of some \mathcal{L}_μ formula, and [19] shows that all ordinals below ω^ω are closure ordinals for some formula in the two-way μ -calculus, i.e. the extension of \mathcal{L}_μ by backwards modalities. The situation in the intuitionistic setting is studied in e.g. [11].

The present paper is concerned with a slightly different but related question: we ask for closure ordinals on particular structures, i.e. at which ordinal do the iteration processes of *all* \mathcal{L}_μ formulas stabilise? Similar problems have been investigated by Barwise and Moschovakis in the context of first-order logic, see e.g. [4]. The problem of finding closure ordinals of classes of structures relates to the previous problem, since the former closure ordinals obviously bound the latter over the given class. Hence, studying closure ordinals of classes of structures contributes to the understanding of closure ordinals of formulas.

For \mathcal{L}_μ , the cardinality of the structure in question is an obvious upper bound for its closure ordinal. Hence, on finite structures, fixpoint iteration must necessarily stabilise at some finite bound that is uniform for all formulas. This simple observation extends to structures with a *finite bisimulation quotient*, as an immediate consequence of \mathcal{L}_μ 's inability to distinguish bisimilar states. An interesting question arises as the converse of this: does an inherently infinite structure, i.e. one whose bisimulation quotient is infinite, allow formulas to have an infinite fixpoint iteration process? Put differently, are there structures with an infinite bisimulation quotient such that fixpoint iteration for *any* \mathcal{L}_μ formula converges after finitely many steps? It is tempting to equate having a finite bisimulation quotient with the finite convergence of all \mathcal{L}_μ fixpoints, yet the answer to the latter question is “yes”.

We construct an LTS – in fact, an infinite word w_∞ – which has an infinite bisimulation quotient but all \mathcal{L}_μ formulas are equivalent to some finite approximation over it. Locally it seems to be regular, and no \mathcal{L}_μ formula can “see” the non-regular global pattern in it. Hence, \mathcal{L}_μ fixpoints cannot exploit this non-regularity in order to only stabilise after more than finitely many iteration steps. Local regularity means that w_∞ is self-similar: it is made of building blocks of increasing size, and some postfixes of it are w_∞ again if one maps suitable building blocks to certain symbols in the word's alphabet.

The result on finite convergence over w_∞ is obtained by first reducing the question for arbitrary \mathcal{L}_μ formulas to that of alternation-free ones. The aforementioned alternation-hierarchy collapse cannot simply be used off-the-shelf here as it makes no statement about the preservation of (in-)finiteness of closure ordinals. We then reduce the question to that for \mathcal{L}_μ formulas only containing fixpoints of one sort. We transform this into the analysis of runs of a very rudimentary fragment of alternating parity automata over w_∞ , exploiting its self-similar structure. Decidability of \mathcal{L}_μ model checking over w_∞ follows as a corollary.

Finally, we show that the reason for the discrepancy between infinite bisimulation quotients and infinite \mathcal{L}_μ closure ordinals is to be found in the regularity of \mathcal{L}_μ 's expressive power. We show that there are formulas of HFL – a natural higher-order extension of \mathcal{L}_μ [22] – that do

not have finite convergence on w_∞ . This is interesting because HFL can define the Kleene fixpoint iteration for \mathcal{L}_μ [8], and this can be used to disentangle fixpoint alternation, albeit at the cost of blow-up in formula size and type order, and the restriction to structures on which each \mathcal{L}_μ formula is equivalent to some finite approximation. Thus, the result of the paper at hand implies that this is not necessarily restricted a priori to structures with finite bisimulation quotients, as the aforementioned w_∞ is a counterexample.

The paper is organised as follows. In Sect. 2 we recall necessary preliminaries. In Sect. 3 we take a detailed look at fixpoint iteration for \mathcal{L}_μ formulas to make the notion of “finite convergence” formal. In Sect. 4 we develop an automata-theoretic criterion for a class of words to have finite convergence. We use this in Sect. 5 to prove finite convergence for w_∞ . Sect. 6 contains the considerations on modal fixpoints of higher-order on w_∞ . Sect. 7 concludes with remarks on further work.

2 Preliminaries

Words and languages. An *alphabet* is a finite, nonempty set Σ of *letters*, denoted by a, b, \dots . A Σ -word is a finite or infinite sequence of letters. Infinite words are also called ω -words. The empty word is denoted by ε . We write $w = a_1 a_2 \dots a_n$ for finite words, and $w = a_1 a_2 \dots$ for infinite words. Given two finite words u and v with $v \neq \varepsilon$, then $u \cdot v^\omega$ is an infinite word. In both the finite and the infinite case, $w[i]$ denotes the i -th letter of w . Σ^* is the set of finite Σ -words, a subset of which is a Σ -*language* (of finite words). Languages of ω -words have no role in this paper. If the alphabet is clear from context, we simply speak of words and languages.

Words are a special case of labelled transition systems for which the notion of bisimilarity is well-known. We can simplify the definition for word structures and call two positions in a word *bisimilar* if they have the same postfix, i.e. if the rest of the word is the same from both positions. Obviously, an ω -word with two bisimilar positions is ultimately periodic. The bisimulation quotient of a word is the quotient w.r.t. bisimilarity. It is either the word itself (when all positions are mutually non-bisimilar), or a lasso-shaped representation of it (when at least two, and then necessarily all following pairs of positions are bisimilar).

The modal μ -calculus. We introduce the modal μ -calculus in its linear-time version only. Let Σ be an alphabet, let \mathcal{X} be a set of fixpoint variables. The syntax of the linear-time μ -calculus in negation normal form, just \mathcal{L}_μ from now on, is given by the grammar

$$\varphi ::= a \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid X \mid \mu X. \varphi \mid \nu X. \varphi$$

where $X \in \mathcal{X}$ and $a \in \Sigma$. Other connectives such as \mathbf{tt} , \mathbf{ff} , \rightarrow etc. are defined as usual. Note that negation is definable using De Morgan, $\neg a \equiv \bigvee_{b \neq a} b$ and duality between μ and ν .

The notion of a subformula is standard. The *size* $|\varphi|$ of a formula φ is the number of its distinct subformulas. Fixpoint quantifiers $\sigma \in \{\mu, \nu\}$ act as variable binders. The notion of free and bound occurrence, as well as that of a closed formula are as usual. In a formula $\sigma X. \varphi$, the subformula φ is the *defining* formula of X . A variable bound by μ is a *least-fixpoint variable*. It is a *greatest-fixpoint variable* if it is bound by ν .

We assume formulas to be *well-named* in the sense that each fixpoint variable is bound at most once. Clearly, any formula is equivalent to a well-named one via renaming of variables. In a well-named formula φ , there is a function fp_φ that maps each fixpoint variable to the defining formula of this variable. We drop the index if φ is clear from context. Well-namedness induces a partial order $<_{\text{fp}}$ defined via $X <_{\text{fp}} Y$ iff $\text{fp}_\varphi(X)$ is a proper subformula of $\text{fp}_\varphi(Y)$. Note that, if $X <_{\text{fp}} Y$, then X has no free occurrences in $\text{fp}_\varphi(Y)$.

24:4 Finite Convergence of μ -Calculus Fixpoints on Genuinely Infinite Structures

We call a formula *unipolar* if it only contains one kind of fixpoint quantifiers. We call a formula *alternation-free* if no variable bound by a least fixpoint quantifier appears freely in the defining formula of a greatest fixpoint quantifier, and vice versa. In alternation-free formulas, the fixpoint variables can be partitioned into sets $\mathcal{X}_1, \dots, \mathcal{X}_k$ such that

- fixpoint variables do not appear freely in the defining formulas of variables from another set of the partition,
- all variables of one set have the same polarity, and
- for all $X \in \mathcal{X}_i$ and $Y \in \mathcal{X}_j$ with $i \neq j$: if $X <_{\text{fp}} Y$ then $j < i$.

A closed formula that contains no fixpoint quantifiers is in Basic Modal Logic (ML). The notion of *modal depth* $\text{md}(\varphi)$ of φ is defined as usual: $\text{md}(a) = 0$, $\text{md}(\psi_1 \vee \psi_2) = \text{md}(\psi_1 \wedge \psi_2) = \max\{\text{md}(\psi_1), \text{md}(\psi_2)\}$ and $\text{md}(\bigcirc \psi) = 1 + \text{md}(\psi)$.

Let $\eta: \mathcal{X} \rightarrow 2^{\mathbb{N}}$ be an *environment*. The semantics of an \mathcal{L}_μ formula on an ω -word w is a set of positions defined inductively as follows:

$$\begin{aligned} \llbracket a \rrbracket_\eta^w &= \{i \in \mathbb{N} \mid w[i] = a\} & \llbracket X \rrbracket_\eta^w &= \eta(X) \\ \llbracket \varphi \wedge \psi \rrbracket_\eta^w &= \llbracket \varphi \rrbracket_\eta^w \cap \llbracket \psi \rrbracket_\eta^w & \llbracket \mu X. \varphi \rrbracket_\eta^w &= \bigcap \{U \subseteq \mathbb{N} \mid \llbracket \varphi \rrbracket_{\eta[X \mapsto U]}^w \subseteq U\} \\ \llbracket \varphi \vee \psi \rrbracket_\eta^w &= \llbracket \varphi \rrbracket_\eta^w \cup \llbracket \psi \rrbracket_\eta^w & \llbracket \nu X. \varphi \rrbracket_\eta^w &= \bigcup \{U \subseteq \mathbb{N} \mid U \subseteq \llbracket \varphi \rrbracket_{\eta[X \mapsto U]}^w\} \\ \llbracket \bigcirc \varphi \rrbracket_\eta^w &= \{i \in \mathbb{N} \mid i + 1 \in \llbracket \varphi \rrbracket_\eta^w\} \end{aligned}$$

We say that φ is *satisfiable* under η over w if $\llbracket \varphi \rrbracket_\eta^w \neq \emptyset$. It is *valid*, written $\models \varphi$, if $\llbracket \varphi \rrbracket_\eta^w = \mathbb{N}$ for all w and η .

Two closed \mathcal{L}_μ formulas φ and ψ are *equivalent*, written $\varphi \equiv \psi$, if they define the same set on all words. We write $\varphi \equiv_w \psi$ to denote that φ and ψ define the same set on w , and $\varphi \equiv_{\mathcal{C}} \psi$ for a class of words \mathcal{C} if φ and ψ define the same set on all words in \mathcal{C} .

Trivial Automata. A *trivial* Σ -automaton (TrA) has the form (Q, δ, q_I, F, b) where

- Q is a finite nonempty set of states with initial state $q_I \in Q$ and final states $F \subseteq Q$,
- $\delta: (Q \setminus F) \times \Sigma \rightarrow Q$ is the transition function and
- $b \in \{0, 1\}$.

A *run* of \mathcal{A} starting from some position i in some ω -word w is a finite or infinite sequence q_0, q_1, \dots of states with $q_0 = q_I$ and $q_{j+1} = \delta(q_j, w_{j+i})$. Note that if such a sequence is finite, then the last state must necessarily be in F since δ is total on $Q \setminus F$. A run is *accepting* if it is finite and $b = 1$, or if it is infinite and $b = 0$. The set of positions in an ω -word defined by \mathcal{A} is the set of positions from which it has an accepting run. Two TrA are equivalent if they define the same set on every infinite Σ -word.

We write $q \xrightarrow{v} q'$ for $q, q' \in Q, v \in \Sigma^*$ to denote that \mathcal{A} will be in state q' after reading the finite word v starting from q . This includes the case in which \mathcal{A} does not read the entirety of v since it stops beforehand. In this case, $q' \in F$.

Trivial automata are reminiscent of DFA but they operate on infinite words. They can be thought of as restricted parity automata with a single priority b . If $b = 1$ then no infinite run is accepting. Acceptance is typically still possible by hitting a state and alphabet symbol to which the transition function assigns \mathfrak{tt} (as a Boolean combination of states). By making acceptance explicit through final states instead, we can define these trivial automata to be deterministic which is useful in proofs later on.

► **Lemma 1.** *Any unipolar \mathcal{L}_μ formula is equivalent to a TrA.*

Proof. (Sketch) It is well-known that \mathcal{L}_μ formulas on words can be translated into alternating parity automata (APA) [23]. Unipolar formulas result in APA of a single priority: 1 for least, 0 for greatest fixpoints. Since the APA has only one priority, determinisation is possible through a double powerset construction, paying attention to states with no successors by either introducing a sink state, or by making them final. ◀

3 Unfolding of Fixpoint Formulas

Single fixpoints. We formalise the notion of finite fixpoint convergence. First consider the case of a formula with a single fixpoint. Let $\mu X.\varphi$ be a formula, η be an environment, and w be a word. Then φ defines a monotonic function $f: T \mapsto \llbracket \varphi \rrbracket_{\eta[X \mapsto T]}^w$. Approximations to the least fixpoint are defined via

$$T_X^0 = \emptyset, \quad T_X^{i+1} = f(T_X^i) = \llbracket \varphi \rrbracket_{\eta[X \mapsto T_X^i]}^w, \quad T_X^\omega = \bigcup_{i \in \mathbb{N}} T_X^i.$$

Since we restrict ourselves to word structures with no branching, we do not need to consider approximations beyond ω , and by Kleene's Fixpoint Theorem we have $T_X^\omega = \llbracket \mu X.\varphi \rrbracket_\eta^w$.

Note that the T_X^i are definable in \mathcal{L}_μ by (renaming instances of) formulas φ^i via $\varphi^0 = \mathbf{ff}$, $\varphi^{i+1} = \varphi[\varphi^i/X]$ independently of w and η . We call φ^i the *i th unfolding* of $\mu X.\varphi$. If over w we have $T_X^i = T_X^\omega$ for some i then the fixpoint is equivalent to its i th unfolding, and the set defined by it can also be defined by a formula without the fixpoint. The definitions for greatest fixpoints are analogue, except that one starts with \mathbb{N} instead of \emptyset , and with \mathbf{tt} instead of \mathbf{ff} .

Multiple and nested fixpoints. For formulas containing more than one fixpoint subformula, possibly in a nested way, it is less clear what it means for these subformulas to be “unfolded n times”, as there is some ambiguity w.r.t. the order in which the participating formulas are to be unfolded. The literature also contains no standard agreed-upon definition. Instead, one of the following three constructions is employed as the respective authors see fit: Unfolding bottom-up, unfolding top-down, or unfolding on demand, following a construction seen in e.g. [21]. We review all three of them here and show that they produce the same formula (cf. Lem. 7). Hence, using either of them as suitable is permissible, and the results obtained in Sect. 5 later on hold for any of these reasonable interpretations of finite fixpoint convergence.

This thorough discussion is necessitated by two reasons: Note that we have very strict requirements with respect to unfolding procedures in the sense that, for an “ n th unfolding”, whenever a fixpoint is to be unfolded during the procedure, it is unfolded exactly n times. This differs from other notions where the amount of unfoldings can vary from fixpoint to fixpoint as long as some fixpoint-free formula is produced. Moreover, contrary to the case of only one fixpoint, monotonicity of the unfolding can be lost for formulas that are not unipolar, contradicting what one might intuitively assume (cf. Ex. 8), in particular w.r.t. stabilisation of the process.

The first unfolding procedure is quite straightforward: Pick a formula that is minimal w.r.t. $<_{\text{fp}}$ and unfold it n times. Clearly, this procedure terminates after k steps, if the formula in question contains k fixpoint definitions. However, it is not immediately clear whether a common n exists if one is interested in producing a formula equivalent to the original one over some structure.

► **Definition 2** (Bottom-up unfolding). Define $\hat{\mu} := \mathbf{ff}$ and $\hat{\nu} := \mathbf{tt}$. Let $n \geq 0$ and let φ be an \mathcal{L}_μ formula. Let X_1, \dots, X_k be an enumeration of its fixpoint variables such that X_i is bound by some $\sigma X_i.\psi_i$, and $X_i \not\prec_{\text{fp}} X_j$ for $j > i$. Let $\varphi_k^n, \dots, \varphi_0^n$ with $\varphi_k^n = \varphi$ be a sequence of formulas defined via

$$\psi_i^0 = \hat{\sigma}_i, \quad \psi_i^{j+1} = \text{fp}_{\varphi_i^n}(X_i)[\psi_i^j/X_i], \quad \varphi_{i-1}^n = \varphi_i^n[\psi_i^n/\sigma_i X_i. \text{fp}_{\varphi_i^n}(X_i)]$$

Then φ_0^n is the n th bottom-up unfolding of φ .

► **Example 3.** Let $\varphi = \nu X. \bigcirc(\mu Y.X \wedge Y)$ and let $n = 2$. Clearly $Y <_{\text{fp}} X$. Hence, $\psi_2^2 = X \wedge (X \wedge \mathbf{ff})$ whence $\varphi_1^2 = \nu X. \bigcirc(X \wedge (X \wedge \mathbf{ff}))$. Moreover $\psi_1^1 = \bigcirc(\mathbf{tt} \wedge (\mathbf{tt} \wedge \mathbf{ff}))$ and, hence, $\varphi_0^2 = \bigcirc(\bigcirc(\mathbf{tt} \wedge (\mathbf{tt} \wedge \mathbf{ff})) \wedge (\bigcirc(\mathbf{tt} \wedge (\mathbf{tt} \wedge \mathbf{ff})) \wedge \mathbf{ff}))$.

The second procedure is perhaps the most straightforward one: Given some formula that contains fixpoints, and some n , pick some fixpoint definition that is maximal w.r.t. $<_{\text{fp}}$ and unfold it n times. Given that this may duplicate fixpoint definitions that are smaller w.r.t. $<_{\text{fp}}$, this requires renaming and raises questions regarding termination of the procedure. Moreover, if one is interested into producing equivalent formulas over some structure, it is, again, not immediately clear whether a common n exists that can be used for all fixpoint definitions simultaneously. We start with an example.

► **Example 4.** Consider again the formula $\nu X. \bigcirc(\mu Y.X \wedge Y)$. Let $\psi = \bigcirc(\mu Y.X \wedge Y)$. By unfolding X twice as per above, we obtain the sequence of formulas $\psi^0 = \mathbf{tt}, \psi^1 = \bigcirc(\mu Y.\mathbf{tt} \wedge Y), \psi^2 = \bigcirc(\mu Y.(\bigcirc(\mu Y.\mathbf{tt} \wedge Y)) \wedge Y)$. Not only are there now two fixpoint definitions involving Y , their variables are also comparable via $<_{\text{fp}}$. However, by renaming one of them, we obtain the formula $\bigcirc(\mu Y.(\bigcirc(\mu Y'.\mathbf{tt} \wedge Y')) \wedge Y)$. Note that the two variables are not mutually recursive since Y does not appear in the defining formula of Y' .

Since the two variables are comparable, and $Y >_{\text{fp}} Y'$, we proceed by unfolding it which, after renaming, results in $\bigcirc((\bigcirc(\mu Y'.\mathbf{tt} \wedge Y')) \wedge ((\bigcirc(\mu Y''.\mathbf{tt} \wedge Y'')) \wedge \mathbf{ff}))$.

Again, we obtain two fixpoint definitions. However, this time, the two variables in question are not comparable via $<_{\text{fp}}$, whence the order of unfolding is obviously not important. We unfold Y' first and obtain $\bigcirc((\bigcirc(\mathbf{tt} \wedge (\mathbf{tt} \wedge \mathbf{ff}))) \wedge ((\bigcirc(\mu Y''.\mathbf{tt} \wedge Y'')) \wedge \mathbf{ff}))$ and then, after unfolding Y'' , the formula $\bigcirc((\bigcirc(\mathbf{tt} \wedge (\mathbf{tt} \wedge \mathbf{ff}))) \wedge ((\bigcirc(\mathbf{tt} \wedge (\mathbf{tt} \wedge \mathbf{ff}))) \wedge \mathbf{ff}))$.

This is the same formula as the one obtained by bottom-up unfolding in Ex. 3. This is in fact no coincidence, cf. Lemma 7 below.

► **Definition 5** (Top-down unfolding). Let $n \geq 0$ and let φ be an \mathcal{L}_μ formula. Define a sequence $\varphi_0^n, \varphi_1^n, \dots$ where $\varphi_0^n = \varphi$, and φ_{i+1}^n is obtained from φ_i^n via the following process: if φ_i^n contains no fixpoint definitions, $\varphi_{i+1}^n = \varphi_i^n$. Otherwise, let X be a variable that is maximal w.r.t. $<_{\text{fp}}$ in φ_i^n . Let $\sigma X.\psi$ be the subformula that defines X . Define $\psi^0 = \hat{\sigma}$ and $\psi^{j+1} = \psi[\psi^j/X]$. Then $\varphi_{i+1}^n = \varphi_i^n[\psi^{i+1}/\sigma X.\psi]$ where ψ^{i+1} is a copy of ψ^i made well-named via renaming of variables. If $\varphi_i^n = \varphi_{i+1}^n$, then the n th top-down unfolding of φ is φ_i^n .

As already said, is not immediately obvious that the above process terminates, but Ex. 4 already gives a hint. Unfolding a fixpoint formula may duplicate other, inner fixpoint formulas but the duplicates are independent of each other. Unfolding the outer may create further duplicates of inner duplicates, but these are not mutually recursive, which gives a termination argument.

In order to not deal with ambiguities around the termination of the process, and to avoid issues around unfolding formulas containing free fixpoint variables (cf. the bottom-up approach), we review a third definition of the n th unfolding of a fixpoint formula centered

around tracking for each fixpoint how often it has been unfolded already. This procedure is also folklore and based on the well-known notion of μ -signatures [21] or techniques used to unfold parity automata into \mathcal{L}_μ -formulas [7].

► **Definition 6.** Let $n \geq 0$ and let φ be an \mathcal{L}_μ formula. Let X_1, \dots, X_k be an enumeration of its fixpoint variables such that $X_i \not\prec_{\text{fp}} X_j$ for $j > i$ and such that σ_i denotes the (polarity of the) fixpoint quantifier for X_i . For a tuple $s = (c_1, \dots, c_k)$ let $s(i) = c_i$ if $1 \leq i \leq k$ and, if $s(i) > 0$, define $s[i--]$ as the k -tuple $(c_1, \dots, c_i - 1, n, \dots, n)$.

Define φ^n as φ^{s_I} , where $s_I = (n, \dots, n)$ and ψ^s is given inductively as

$$\begin{aligned} a^s &= a & (\psi_1 \vee \psi_2)^s &= \psi_1^s \vee \psi_2^s \\ (\neg\psi)^s &= \neg\psi^s & (\psi_1 \wedge \psi_2)^s &= \psi_1^s \wedge \psi_2^s \\ (\bigcirc\psi)^s &= \bigcirc\psi^s & X_i^s = (\sigma X_i. \text{fp}_\varphi(X_i))^s &= \begin{cases} \hat{\sigma}_i & , \text{ if } s[i] = 0 \\ \text{fp}_\varphi(X_i)^{s[i--]} & , \text{ otherwise.} \end{cases} \end{aligned}$$

Clearly, φ^{s_I} is well-defined and fixpoint-free. Well-definedness follows from the fact that $s[i--]$ is smaller than s in the lexicographical ordering. Note that we do not have to deal with well-namedness since no intermediate formulas containing fixpoint definitions occur due to the inductive definition centered around s .

We now establish that all three definitions given above actually produce the same formulas:

► **Lemma 7.** Let φ be an \mathcal{L}_μ formula. Then the bottom-up unfolding of φ (cf. Def. 2) and the top-down unfolding of φ (cf. Def. 5) are equivalent to the unfolding defined in Def. 6. In particular, the top-down unfolding is well-defined.

The proof has been moved to the appendix. It mostly consists of tracking the various substitutions.

We say that φ is equivalent to its n th unfolding over some word w if $\varphi \equiv_w \varphi^n$ for all $m \geq n$, i.e. if φ^n defines the same set on each of these words, and so do all further unfoldings. Note that, contrary to the case of a single fixpoint variable, it is not automatically the case that if $\varphi^n \equiv_w \varphi$, then $\varphi^{n+1} \equiv_w \varphi$. To illustrate this, consider the following example:

► **Example 8.** Let $\varphi = \nu X. \mu Y. (a \wedge \bigcirc X) \vee \bigcirc Y$. It defines the set of all positions after which a occurs infinitely often. Its first unfoldings are

$$\begin{aligned} \varphi^0 &= \text{tt} \\ \varphi^1 &= (a \wedge \bigcirc \text{tt}) \vee \bigcirc \text{ff} \equiv a \\ \varphi^2 &= (a \wedge \bigcirc((a \wedge \bigcirc \text{tt}) \vee \bigcirc((a \wedge \bigcirc \text{tt}) \vee \bigcirc \text{ff}))) \\ &\quad \vee \bigcirc((a \wedge \bigcirc((a \wedge \bigcirc \text{tt}) \vee \bigcirc((a \wedge \bigcirc \text{tt}) \vee \bigcirc \text{ff}))) \vee \bigcirc \text{ff}) \\ &\equiv (a \wedge \bigcirc(a \vee \bigcirc a)) \vee \bigcirc((a \wedge \bigcirc(a \vee \bigcirc a))) \end{aligned}$$

Take $w = (ba)^\omega$. Then φ^0 obviously defines \mathbb{N} , while φ^1 defines $\{2n+1 \mid n \in \mathbb{N}\}$ and then φ^2 again defines \mathbb{N} and so do all further approximations. Similar examples can be constructed to separate any two approximations. In fact, over a word of the form $b^1 a b^2 a b^3 a \dots$, the formula φ is not equivalent to any of its unfoldings φ^i with $i \geq 1$, but still defines \mathbb{N} .

Note that φ from Ex. 8 is not unipolar. For unipolar formulas, monotonicity can be used to show that $\models \varphi^i \rightarrow \varphi^{i+1}$ for least fixpoint formulas, resp. $\models \varphi^{i+1} \rightarrow \varphi^i$ for greatest fixpoint formulas holds for all i .

4 Finite Fixpoint Convergence for \mathcal{L}_μ

In this section we define the notion of a word having finite convergence, i.e. the property that all formulas, or all formulas of a certain kind, are equivalent to a finite unfolding over this word. We also develop a sufficient criterion in terms of runs of TrA, for this to hold.

► **Definition 9.** *Let w be an infinite word and let Φ be a set of \mathcal{L}_μ formulas. We say that w has finite convergence for Φ if, for every $\varphi \in \Phi$, there is n such that φ is equivalent to φ^n over w . We say that w has finite convergence for \mathcal{L}_μ , if the above holds for the set of all \mathcal{L}_μ formulas.*

The rest of the section is devoted to reducing finite convergence over w for the set of \mathcal{L}_μ formulas to a rather simple criterion on the runs of TrA over w . Lemmas 10 and 11 establish that, if a word has finite convergence for the set of alternation-free formulas, it also has finite convergence for the set of all \mathcal{L}_μ formulas. The rest of the section establishes a criterion for a word to have finite convergence for the set of alternation-free formulas.

► **Lemma 10.** *Let w be an infinite word that has finite convergence for the set of all alternation-free \mathcal{L}_μ formulas. Then, every closed \mathcal{L}_μ formula φ of the form $\sigma X.\psi$ is equivalent over w to one in ML, and so are all its approximations X^i . Moreover, there is some i such that φ agrees with X^i over w .*

Proof. Let w and $\varphi = \mu X.\psi$. The case of $\sigma = \nu$ is analogous. By [15], φ is equivalent to an alternation-free formula over the class of all words. Then, by the assumption of the lemma, there is $\varphi' \in \text{ML}$ that is equivalent to this alternation-free formula, obtained via some finite unfolding of φ . Now let X^i be the i th approximation of φ and let ψ_i be the \mathcal{L}_μ formula that defines it. Note that it possibly contains fixpoint definitions, since the only fixpoint to be unfolded is X . However, with the same argument as before we obtain that ψ_i also must be equivalent to some alternation-free formula and, hence, to some formula ψ'_i in ML.

Regarding the claim that one of the approximations is already equivalent to φ , assume that this is not the case. Since the ψ'_i are obtained as formulas equivalent to finite approximations of φ , we must have that for each $i \in \mathbb{N}$, there must be some position $j_i \in \llbracket \psi'_{i+1} \rrbracket^w \setminus \llbracket \psi'_i \rrbracket^w$ and, hence $j_i \in \llbracket \varphi' \rrbracket^w \setminus \llbracket \psi'_i \rrbracket^w$. Consider the set $\Phi = \{\varphi'\} \cup \{\neg\psi'_i \mid i \in \mathbb{N}\}$. We show that it is satisfiable using the Compactness Theorem for ML. Consider any finite subset Ψ of Φ , w.l.o.g. it is of the form $\{\varphi'\} \cup \{\neg\psi'_i \mid i \leq k\}$ for some k . By the above, Ψ is satisfiable by a postfix of w , starting at j_k . Hence, Φ is also satisfiable, i.e. there is an ω -word w' that satisfies φ' , but none of the ψ'_i . This is a contradiction, since $\llbracket \varphi' \rrbracket^{w'} = \bigcup_{i \in \mathbb{N}} \llbracket \psi'_i \rrbracket^{w'}$ by definition. This contradiction stems from the assumption that there is not already some i such that $\psi'_i \equiv_w \varphi' \equiv_w \varphi$. This finishes the proof. ◀

► **Lemma 11.** *Let w be an infinite word. If w has finite convergence for the set of all alternation-free \mathcal{L}_μ formulas, it has finite convergence for the set of all \mathcal{L}_μ formulas.*

Proof. Let φ be an \mathcal{L}_μ formula. Using Lem. 10, we can obtain a non-uniform unfolding φ' of φ that is equivalent to φ over w , i.e. we show that there is m such that, following the pattern of the top-down unfolding procedure in Def. 5, for each fixpoint subformula there is some $n \leq m$ such that unfolding it n times yields an equivalent subformula. In a second step, we show that we also obtain a formula equivalent to φ if we unfold all fixpoint subformulas exactly m times. This is not immediately obvious due to the non-monotonicity seen in Ex. 8.

Towards the first goal, define a sequence $\varphi_0, \varphi_1, \dots$ where $\varphi_0 = \varphi$, and φ_{i+1} is obtained from φ_i via the following process, similarly to the top-down unfolding: if φ_i contains no fixpoint definitions $\varphi_{i+1} = \varphi_i$. Otherwise, let X be a variable in φ_i that is maximal w.r.t

$<_{\text{fp}}$. Let $\sigma X.\psi$ be the subformula that defines X . By Lem. 10, there is some m_i such that the m_i th unfolding of ψ is equivalent to $\sigma X.\psi$, defined via $\psi^0 = \hat{\sigma}$ and $\psi^{i+1} = \psi[\psi^i/X]$. Then $\varphi'_{i+1} = \varphi_i[\psi^{m_i+1}/\sigma X.\psi]$ resulting from the $(m_i + 1)$ th unfolding of X . Let φ_{i+1} be a obtained from φ'_{i+1} via renaming such that φ_{i+1} is well-named. Note that the amount of times each fixpoint is unfolded varies. This is where the above differs from the top-down unfolding. The above process stabilises for the same reason the top-down unfolding is well-defined: unfolding an outermost fixpoint formula will create closed formulas, i.e. the ψ^i as described above are all closed. Hence, while unfolding an outermost fixpoint X can duplicate fixpoints smaller than X w.r.t $<_{\text{fp}}$, the defining formulas of the duplicates reside in different instances of the ψ^i and, hence, are not mutually recursive. In particular, unfolding a formula in ψ^{i+1} may create further duplicates by replicating ψ^i , but since ψ^i is closed, these further duplicates can then be unfolded independently of each other.

Hence, let φ' be the formula that results once this process stabilises. Note that, however, the unfolding is not necessarily uniform. Let $m = 1 + \max\{m_i \mid i \in \mathbb{N}\}$. Since the process stabilises, this maximum exists. We claim that φ is equivalent to φ^m over w . We show this by unfolding φ using the top-down procedure. Note that above, we have established that, for each fixpoint X in the process, there is some $m_i < m$ such that unfolding it $m_i + 1$ times, once it is X 's turn, results in a formula equivalent to the one before. We now show that this property is kept if we instead unfold it m times. Let X be such a variable, and assume that the property holds so far. Note that X must be outermost by now. Let $\sigma X.\psi$ be the defining formula of X . We compare ψ^{m_i} and ψ^m , which are equivalent due to Lem. 10. Note that ψ^{m_i+1} and ψ^{m_i} are equivalent due to the definition of m_i . Since $m \geq m_i$, we have that $\psi^m = \psi^{m_i+k}$ for some k , and it contains ψ^{m_i} as a subformula. Since that subformula is closed, clearly the invariant holds for all fixpoint definitions in ψ^{m_i} , since the process inside this subformula will play out exactly like before. If $m_i + 1 = m$, we are done. Otherwise, consider a subformula in $\psi^{m_i+1+k'}$ for some $k' \leq m - m_i - 1$, but not in ψ^{m_i+1} , i.e., it is in the extra part of the formula due to the extra unfolding. Since $\psi^{m_i+1} \equiv_w \psi^{m_i}$ by definition, we also have that $\psi^{m_i+k'} \equiv_w \psi^{m_i}$. In other words, the part of the formula where X used to be, but some ψ^j has been substituted, is equivalent over w due to the definition of m_i . Moreover, both substituted formulas are closed and, hence, can be exchanged without interfering with unfolding of fixpoint formulas. Hence, for the purposes of fixpoint unfolding, all fixpoint formulas in ψ^m , but not in ψ^{m_i+1} behave like a fixpoint formula in ψ^{m_i+1} , but not in ψ^{m_i} . Since the invariant holds for the latter, it must also hold for the former.

It follows that we can make the unfolding uniform by just using the top-down unfolding process with m . Moreover, any $m' \geq m$ yields the same result by the same reasoning. This finishes the proof. \blacktriangleleft

► **Remark 12.** Note that Lem. 11 yields more than just the collapse of \mathcal{L}_μ to ML over w which can already be inferred from the collapse of \mathcal{L}_μ to alternation-free \mathcal{L}_μ over the class of all words (see [15]). Lem. 11 yields that every \mathcal{L}_μ formula φ is equivalent, over w , not only to some ML formula, but one obtained as an unfolding of φ (cf. also the remarks after Def. 9).

► **Lemma 13.** *Let w be an infinite word. If w has finite convergence for the set of all unipolar \mathcal{L}_μ formulas, it has finite convergence for the set of all alternation-free \mathcal{L}_μ formulas.*

The proof is a standard induction on the alternation classes using the fact that alternation-free \mathcal{L}_μ is obtained by capture-avoiding substitution of unipolar formulas. It is spelled out in the appendix.

24:10 Finite Convergence of μ -Calculus Fixpoints on Genuinely Infinite Structures

► **Definition 14.** Let w be an ω -word and \mathcal{A} be a TrA. We say that \mathcal{A} has k -bounded runs on w if all finite runs of \mathcal{A} in w are of length k or less. We say that TrA have bounded runs on w if, for every TrA \mathcal{A} , there is k such that \mathcal{A} has k -bounded runs on w .

Clearly, if \mathcal{A} has k -bounded runs on w , then acceptance of \mathcal{A} can be expressed by a ML-formula of modal depth at most k , i.e., there is some ML-formula φ of modal depth at most k such that $\llbracket \varphi \rrbracket^w = \{i \mid \mathcal{A} \text{ accepts from } i\}$.

► **Lemma 15.** Let w be an infinite word. If TrA have bounded runs over w then w has finite convergence for all unipolar formulas.

Proof. We only show the result for least fixpoint formulas; for greatest it is analogous. Let w be given, φ be unipolar containing only least fixpoints, and φ^i be its i th unfolding. Since φ is unipolar, it is equivalent to a TrA \mathcal{A} by Lem. 1. By the assumption, there is k such that if \mathcal{A} halts on w from some position then it halts in k steps or less. Hence, acceptance of \mathcal{A} on w can be expressed by some ML formula ψ , which means that ψ is equivalent to φ over w . Then, similar to the proof of Lem. 10, we can use the Compactness Theorem to obtain that φ is already equivalent to some φ^i . ◀

Lemmas 11, 13 and 15 yield the following.

► **Corollary 16.** Let w be an infinite word. If TrA have bounded runs on w , then w has finite convergence for \mathcal{L}_μ .

5 A Word with Finite Convergence

We are now ready to give the construction of a word with finite fixpoint convergence. Let $\Sigma = \{a, b\}$. We define w_∞ using the following mutually recursive definitions of families of finite words α_i, β_i :

$$\alpha_0 = a \quad , \quad \beta_0 = b \quad , \quad \alpha_{i+1} = \alpha_i \alpha_i \beta_i \alpha_i \alpha_i \quad , \quad \beta_{i+1} = \beta_i \beta_i \alpha_i \beta_i \beta_i$$

For example, $\alpha_2 = aabaa aabaa bbabb aabaa aabaa$. Then $w_\infty = \alpha_0 \alpha_1 \dots$.

We obtain the following properties of w_∞ :

► **Lemma 17.** Let w_∞ be as above. Then it holds that

1. the length of α_i and β_i is 5^i ,
2. α_i and β_i do not overlap, i.e. the minimal size for a word that contains both of them is $2 \cdot 5^i$,
3. the postfix of w_∞ starting after position $\sum_{j=0}^{i-1} 5^j$, i.e. at the first occurrence of α_i , can be considered a word in $\{\alpha_i, \beta_i\}^\omega$,
4. α_i , respectively β_i occurs at most 4 times in a row before the other one occurs.
5. the distance from any position to the next occurrence of one of α_i or β_i is at most $5^i - 1$,
6. The postfix of w_∞ starting after position $\sum_{j=0}^{i-1} 5^j$ is $h(w_\infty)$ for the homomorphism h with $h(a) = \alpha_i$ and $h(b) = \beta_i$,
7. w_∞ has an infinite bisimulation quotient.

Proof. Item 1 is an immediate consequence of the definition of w_∞ . Item 2 follows from a straightforward induction: The claim is obvious for $i = 0$, and for $i > 0$ we note that any potential overlap of α_{i+1} and β_{i+1} either induces overlap of α_i and β_i , or contradicts the fact that α_{i+1} contains only one occurrence of β_i . Item 3 follows from the construction of α_{i+1} , resp. β_{i+1} . Towards Item 4, we use Item 2 to note that consecutive occurrences of α_i , resp.

β_i must occur aligned to the building pattern of w_∞ , i.e. following the pattern exhibited in Item 3 applied to α_{i+1} and β_{i+1} . The claim then follows directly from the construction of α_{i+1} and β_{i+1} .

Regarding Item 5, note that, by Item 3, eventually, w_∞ consists entirely of a sequence of α_i and β_i . Hence, one must occur after a distance of at most $5^i - 1$. Moreover, since the first occurrence of α_i is at position $\sum_{j=0}^{i-1} 5^j \leq 5^i - 1$, the claim also holds for the initial part of the word. Item 6 follows from Item 3 and the building pattern of w_∞ .

It remains to prove Item 7, i.e. that w_∞ has an infinite bisimulation quotient. This holds since all positions of the form $\sum_{j=0}^{i-1} 5^j$, i.e. the first occurrences of α_i for $i \in \mathbb{N}$, are pairwise not bisimilar. Towards this, note that α_i^3 is the word following at position $\sum_{j=0}^{i-1} 5^j$, since the α_{i+1} following the first occurrence of α_i begins with α_i^2 . Conversely, all positions of the form $\sum_{j=0}^{i'-1} 5^j$ with $i' > i$ mark the beginning of the first $\alpha_{i'}$, which begins with $\alpha_i^2 \beta_i$ by construction. Hence, the positions $\sum_{j=0}^{i-1} 5^j$ and $\sum_{j=0}^{i'-1} 5^j$ for $i' > i$ are not bisimilar, which yields infinitely many pairwise not bisimilar positions. ◀

The aim now is to show that TrA have finite runs on w_∞ . Let $\mathcal{A} = (Q, \delta, q_I, F, b)$ be a TrA, fixed for the remainder of the section. W.l.o.g. $b = 1$ for the remainder of the section, the proof for $b = 0$ is completely symmetric. Consider the subsets $A_0, A_1, \dots \subseteq Q$ and $B_0, B_1, \dots \subseteq Q$ defined via $q \in A_i$ iff $q \xrightarrow{\alpha_i} q'$ for some $q' \in F$ and $q \in B_i$ iff $q \xrightarrow{\beta_i} q'$ for some $q' \in F$.

Clearly, $A_i \subseteq A_{i+1}$ for all $i \geq 0$ since α_{i+1} starts with α_i , whence any word that begins with α_{i+1} also begins with α_i . Moreover, since $|Q| < \infty$, there must be $i, h \in \mathbb{N}$ such that $A_j = A_i$ for all $j \geq i$ and $B_j = B_h$ for all $j \geq h$. Let $k = 1 + \max\{i, h\}$, $A = A_k$ and $B = B_k$. Note that $A \cap B$ can be nonempty, and both A and B can be empty. Let $M = Q \setminus (A \cup B)$. Then M is the set of states such that \mathcal{A} will not have accepted if it reads α_j or β_j for any j .

We now show that the self-similarity (cf. Lem. 17.6) and w_∞ eventually becoming almost featureless from the perspective of a bounded-memory automaton (cf. Lem. 17.3), together imply that a TrA can get trapped in M if it does not escape it fast enough.

► **Lemma 18.** *Let $q \in M$ and $j \geq k$. Then $q \xrightarrow{\alpha_j} q'$ for some $q' \in M$, and $q \xrightarrow{\beta_j} q'$ for some $q' \in M$.*

Proof. Let $q_0 \in M$, $j \geq k$. Remember that $\alpha_j = \alpha_{j-1}^2 \beta_{j-1} \alpha_{j-1}^2$. Note that $q_0 \xrightarrow{\alpha_{j-1}} q_1$ for some $q_1 \notin A$, because otherwise $q_0 \xrightarrow{\alpha_j} q_2$ for some $q_2 \in F$ contradicting $q_0 \in M$. Moreover, $q_0 \xrightarrow{\alpha_{j-1} \alpha_{j-1}} q_2$ for some $q_2 \notin A$. If it were the case that $q_2 \in A$, then there would be q_1 with $q_0 \xrightarrow{\alpha_{j-1}} q_1$ and $q_1 \xrightarrow{\alpha_{j-1}} q_2$. Since $q_2 \in A$, there must be $q_3 \in F$ such that $q_2 \xrightarrow{\alpha_{j-1}} q_3$. Hence, $q_1 \xrightarrow{\alpha_j} q_3$, which implies $q_1 \in A$. This contradicts the previous result, whence $q_2 \notin A$.

In summary, $q \xrightarrow{\alpha_{j-1}} q'$ or $q \xrightarrow{\alpha_{j-1} \alpha_{j-1}} q'$ for a state $q \in M$ implies $q' \notin A$ and it can be easily inferred that $q \xrightarrow{\alpha_{j-1} \alpha_{j-1}} q'$ also implies $q' \notin B$. The same holds symmetrically for β_{j-1} . Now, these findings can be used to prove the lemma. Per assumption, the automaton does not halt reading α_j starting from q_0 . Hence, there are q_1, q_2, q_3 such that $q_0 \xrightarrow{\alpha_{j-1} \alpha_{j-1}} q_1$, $q_1 \xrightarrow{\beta_{j-1}} q_2$ and $q_2 \xrightarrow{\alpha_{j-1} \alpha_{j-1}} q_3$. From the findings above it immediately follows that $q_1 \in M$. With the symmetric arguments for β_j it follows that $q_2 \notin B$ and from the fact that $q_0 \in M$ it also follows that $q_2 \notin A$, whence $q_2 \in M$. Then, with the same arguments as for q_1 and, we obtain that $q_3 \in M$, too. The case for β_j is analogous. ◀

We are now ready to prove that all TrA are bounded over the word w .

► **Theorem 19.** *TrA have bounded runs on w_∞ .*

Proof. Let \mathcal{A} be a TrA, M, k be as above and let l be some position in w_∞ . We show that if \mathcal{A} accepts from position l , then it does so within $6 \cdot 5^k - 1$ many steps. Let l' be the first position after l from which α_k or β_k starts. Let u be the word from position l to position l' . If $q_T \xrightarrow{u} q$ for $q \in F$, we are done since by Lem. 17.5 we have $|u| \leq 5^k - 1$.

Otherwise, let $\gamma_1 \cdots \gamma_5$ be the sequence of length $5 \cdot 5^k$ following l' , which necessarily consists of α_k and β_k by Lem. 17.3. We prove that \mathcal{A} must accept within this sequence. Let q_0, q_1, \dots, q_5 with $q = q_0$ be the sequence of states such that $q_i \xrightarrow{\gamma_{i+1}} q_{i+1}$. Then $q_i \notin M$ for all $0 \leq i \leq 5$, for otherwise, by Lem. 17.3 and Lem. 18, \mathcal{A} does not accept at all from l since the run gets trapped in M , which contradicts the assumption on acceptance from l . By Lem. 17.4 the sequence $\gamma_1 \cdots \gamma_5$ must contain two consecutive α_k followed by β_k or two consecutive β_k followed by α_k . W.l.o.g. suppose that two consecutive α_k are followed by β_k and that this concerns $\gamma_1, \gamma_2, \gamma_3$. If $q_0 \in A, q_1 \in A$ or $q_2 \in B$, we are done, since acceptance follows within the next γ_i . The remaining possibility is that all of $q_0, q_1 \in B \setminus A$ and $q_2 \in A \setminus B$ hold. However, this is not possible: Since $q_1 \xrightarrow{\alpha_k} q_2$ and $q_2 \in A$ implies that there must be $q' \in F$ such that $q_2 \xrightarrow{\alpha_k} q'$, we have that $q_1 \xrightarrow{\alpha_{k+1}} q'$ which implies $q_1 \in A$. Hence, \mathcal{A} either accepts within $l' - l + 5 \cdot 5^k \leq 6 \cdot 5^k - 1$ steps from l , or does not accept from l at all. \blacktriangleleft

Putting this together with the results obtained in the previous section we obtain the following.

► **Corollary 20.** w_∞ has finite convergence for \mathcal{L}_μ (but no finite bisimulation quotient).

This follows from Cor. 16 and Lem. 17.7.

► **Remark 21.** Closer inspection of the proofs in this section yields two additional results. It follows from the proof of Lem. 18 that if $A_i = A_{i+1}$, then $A_i = A_j$ for all $j \geq i$. Hence, k can actually be computed effectively by computing the A_i and the B_i until both sequences stabilise. Moreover, the results of Lem. 18 and Thm. 19 do not rely on the exact form of w_∞ , but rather its pattern, and the results from Sec. 4 do not make any assumptions on the word in question. It is possible to generalise the proof to words constructed via

$$\alpha'_0 = a \quad , \quad \beta'_0 = b \quad , \quad \alpha'_{i+1} = \alpha_i^m \beta_i^m \alpha_i^m \quad , \quad \beta'_{i+1} = \beta_i^m \alpha_i^m \beta_i^m$$

where $m > 1$. I.e. the importance is the symmetry between the α'_i and β'_i , as well as the use of two copies of α'_i at the beginning of α'_{i+1} etc. Moreover, this sequence of finite words does not have to be strictly monotonic in the use of its building blocks, i.e., the result also holds for words of the form $\alpha'_{i_0} \alpha'_{i_1} \cdots$, where $i_j \leq i_{j+1}$ for all $j \geq 0$. The case where the pattern eventually stabilises is not very interesting, of course, but bounded runs for TrA and, hence finite convergence of \mathcal{L}_μ still follow for the case where for all $k \in \mathbb{N}$ there is j such that $i_j \leq k$, i.e., the word uses α'_i of unbounded length as building blocks.

6 Infinite Convergence Through Higher-Order

We now show that the finite convergence of \mathcal{L}_μ formulas on the word w_∞ from Sec. 5 is due to the well-known fact that the expressive power of \mathcal{L}_μ is restricted to regular properties. In contrast, finite convergence does not hold anymore for a higher-order extension of \mathcal{L}_μ with non-regular expressiveness: Higher-Order Modal Fixpoint Logic (HFL). It extends \mathcal{L}_μ by the ability to form function definitions via λ abstraction. We refer to the literature or the appendix for a detailed introduction into HFL [22].

Let $\tau = (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ be the set-theoretic type of functions that consume two functions of type $\mathbb{N} \rightarrow \mathbb{N}$ and return a natural number. Consider the HFL formula

$$\varphi = (\nu(X : \tau). \lambda(f, g : \mathbb{N} \rightarrow \mathbb{N}). f(\mathbf{tt}) \wedge X(f^2 \circ g \circ f^2, g^2 \circ f \circ g^2)) (\langle b \rangle, \langle a \rangle).$$

Here, $\psi = \lambda(f, g : \mathbb{N} \rightarrow \mathbb{N}). f(\mathbf{tt}) \wedge \dots$ defines an anonymous function that takes as input two functions f and g of type $\mathbb{N} \rightarrow \mathbb{N}$ and returns the expression defined by the conjunction. The left conjunct, for example, defines the result of applying f to the set defined by \mathbf{tt} , i.e. \mathbb{N} . The formula $\langle a \rangle$ defines the function $S \mapsto \{i \mid w[i] = a \text{ and } i + 1 \in S\}$, and similarly for $\langle b \rangle$. The fixpoint X itself is now of higher-order, and it is equivalent to the expression $\bigwedge_{i \in \mathbb{N}} \psi^i$, where $\psi^0 = \lambda(f, g : \mathbb{N} \rightarrow \mathbb{N}). \mathbf{tt}$ and $\psi^{i+1} = \psi[\psi^i/X]$. Applying this expression to the original arguments $\langle b \rangle, \langle a \rangle$ in φ and using some β -reduction, we obtain that φ is equivalent to

$$\psi^0(\langle b \rangle, \langle a \rangle) \wedge \psi^1(\langle b \rangle, \langle a \rangle) \wedge \psi^2(\langle b \rangle, \langle a \rangle) \wedge \dots$$

With standard arguments about λ -expressions and modal logic, we obtain that $\psi^i(\langle b \rangle, \langle a \rangle)$ defines the set of positions such that all the α_j for $j < i$ follow, where α_j is as in Sec. 5. Hence, we can conclude that φ defines the set of positions i in w_∞ of Sec. 5 such that the postfix following i starts with α_j for all $j \in \mathbb{N}$.

► **Theorem 22.** *The extension HFL^2 of \mathcal{L}_μ by second-order functions does not have finite convergence on w_∞ .*

Proof. Following the argument above, we get $\llbracket \varphi \rrbracket^{w_\infty} = \emptyset$. However, from the previous analysis we can see that a position $\sum_{i=0}^{j-1} 5^i$, namely the starting point of the first α_j in w_∞ , is still contained in the j th approximant $(\bigwedge_{0 \leq i \leq j} \psi^i(\langle b \rangle, \langle a \rangle))$ but not in the $j + 1$ st one. Hence, φ does not have finite convergence on w_∞ . ◀

► **Remark 23.** It is possible to strengthen Thm. 22 by constructing a formula in HFL^1 , the first-order extension of \mathcal{L}_μ . (We have $\mathcal{L}_\mu = \text{HFL}^0$.) But the construction is complicated and requires further insight into the semantics of HFL, so it is left out for space considerations.

7 Conclusion

We have presented a further contribution to the theory of closure ordinals for μ -calculus, namely the – possibly surprising – fact that having finite bisimulation quotients is not an equivalent but a strictly stronger property than having finite fixpoint convergence, and that this discrepancy is due to \mathcal{L}_μ 's relatively restricted expressive power. As a corollary, we obtain decidability of \mathcal{L}_μ model checking over w_∞ (and the class of words built like it), i.e. for given φ and i , it is decidable whether φ holds at position i in w_∞ . This follows since every \mathcal{L}_μ formula is equivalent to one in ML over w_∞ and it is easily decidable whether the letter at some position is an a or a b . The use of the Compactness Theorem in Lem. 11 might look prohibitively non-constructive, but the result follows from the constructive nature of the proof up to alternation-free \mathcal{L}_μ (cf. Rem. 12). Note that this result does not follow from results around morphic words (cf. e.g. [2]), since w_∞ is not morphic. Hence, the construction of w_∞ can be used as the basis for a new class of infinite structures with decidable \mathcal{L}_μ model checking beyond pushdown processes. However, the design follows a pattern quite similar to morphic words, and we plan to investigate possible links between the two concepts.

There are some other directions into which our research can be extended: for further technical developments the theory of finite convergence has been formulated over classes of structures, even though it has been used here for a single word structure only. It remains

to be seen how far the construction pattern can be stretched, i.e. what a largest class of structures with finite convergence is. One may leave the world of words without losing the ability to reduce from the entire \mathcal{L}_μ to the alternation-free fragment as there are richer classes of structures with corresponding collapse of the alternation hierarchy [14].

The type hierarchy in HFL – which is strict in terms of expressiveness [3] – gives rise to the question whether for each level i , there is a class of structures on which HFL^i has finite convergence but HFL^{i+1} does not. Here, we have answered the question for $i = 0$. For $i > 0$ this is tricky as these higher-order logics lack comparable automata-theoretic support, cf. [6].

References

- 1 B. Afshari and G. E. Leigh. On closure ordinals for the modal μ -calculus. In *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPICs*, pages 30–44. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPICs.CSL.2013.30.
- 2 J.-P. Allouche and J. O. Shallit. *Automatic Sequences - Theory, Applications, Generalizations*. Cambridge University Press, 2003.
- 3 R. Axelsson, M. Lange, and R. Somla. The complexity of model checking higher-order fixpoint logic. *Logical Methods in Computer Science*, 3:1–33, 2007. doi:10.2168/LMCS-3(2:7)2007.
- 4 J. Barwise and Y. N. Moschovakis. Global inductive definability. *J. Symb. Log.*, 43(3):521–534, 1978. doi:10.2307/2273529.
- 5 J. C. Bradfield. The modal μ -calculus alternation hierarchy is strict. In *Proc. 7th Conf. on Concurrency Theory, CONCUR'96*, volume 1119 of *LNCS*, pages 233–246. Springer, 1996. doi:10.1007/3-540-61604-7_58.
- 6 F. Bruse. *Extremal Fixpoints for Higher-Order Modal Logic*. PhD thesis, University of Kassel, Germany, 2020. doi:doi:10.17170/kobra-202008091547.
- 7 F. Bruse, O. Friedmann, and M. Lange. On guarded transformation in the modal μ -calculus. *Log. J. IGPL*, 23(2):194–216, 2015. doi:10.1093/jigpal/jzu030.
- 8 F. Bruse, M. Lange, and É. Lozes. Collapses of fixpoint alternation hierarchies in low type-levels of higher-order fixpoint logic. Proc. Workshop on Programming and Reasoning on Infinite Structures, PARIS'14, 2018.
- 9 M. Czarnecki. How fast can the fixpoints in modal μ -calculus be reached? In *7th Workshop on Fixed Points in Computer Science, FICS 2010, Brno, Czech Republic, August 21-22, 2010*, pages 35–39. Laboratoire d'Informatique Fondamentale de Marseille, 2010. URL: <https://hal.archives-ouvertes.fr/hal-00512377/document#page=36>.
- 10 Mads Dam. Ctl^* and ectl^* as fragments of the modal μ -calculus. *Theor. Comput. Sci.*, 126(1):77–96, 1994. doi:10.1016/0304-3975(94)90269-0.
- 11 S. Ghilardi, M. J. Gouveia, and L. Santocanale. Fixed-point elimination in the intuitionistic propositional calculus. *ACM Trans. Comput. Log.*, 21(1):4:1–4:37, 2020. doi:10.1145/3359669.
- 12 M. J. Gouveia and L. Santocanale. \aleph_1 and the modal μ -calculus. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:1)2019.
- 13 D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ -calculus with respect to monadic second order logic. In *Proc. 7th Conf. on Concurrency Theory, CONCUR'96*, volume 1119 of *LNCS*, pages 263–277. Springer, 1996. doi:10.1007/3-540-61604-7_60.
- 14 J. Gutierrez, F. Klaedtke, and M. Lange. The μ -calculus alternation hierarchy collapses over structures with restricted connectivity. *Theor. Comput. Sci.*, 560:292–306, 2014. doi:10.1016/j.tcs.2014.03.027.
- 15 R. Kaivola. Axiomatizing linear time μ -calculus. In *CONCUR '95: Concurrency Theory, 6th International Conference, Philadelphia, PA, USA, August 21-24, 1995, Proceedings*, volume 962 of *Lecture Notes in Computer Science*, pages 423–437. Springer, 1995. doi:10.1007/3-540-60218-6_32.

- 16 S. C. Kleene. On a notation for ordinal numbers. *Journal of Symbolic Logic*, 3:150–155, 1938.
- 17 D. Kozen. Results on the propositional μ -calculus. *TCS*, 27:333–354, 1983. doi:10.1016/0304-3975(82)90125-6.
- 18 E. Lozes. A type-directed negation elimination. In *Proceedings Tenth International Workshop on Fixed Points in Computer Science, FICS 2015, Berlin, Germany, September 11-12, 2015*, volume 191 of *EPTCS*, pages 132–142, 2015. doi:10.4204/EPTCS.191.12.
- 19 G. C. Milanese and Y. Venema. Closure ordinals of the two-way modal μ -calculus. In *Logic, Language, Information, and Computation - 26th International Workshop, WoLLIC 2019, Utrecht, The Netherlands, July 2-5, 2019, Proceedings*, volume 11541 of *Lecture Notes in Computer Science*, pages 498–515. Springer, 2019. doi:10.1007/978-3-662-59533-6_30.
- 20 M. Otto. Eliminating recursion in the μ -calculus. In *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 531–540. Springer, 1999. doi:10.1007/3-540-49116-3_50.
- 21 R. S. Streeck and E. A. Emerson. The propositional mu-calculus is elementary. In *Automata, Languages and Programming, 11th Colloquium, Antwerp, Belgium, July 16-20, 1984, Proceedings*, volume 172 of *Lecture Notes in Computer Science*, pages 465–472. Springer, 1984. doi:10.1007/3-540-13345-3_43.
- 22 M. Viswanathan and R. Viswanathan. A higher order modal fixed point logic. In *Proc. 15th Int. Conf. on Concurrency Theory, CONCUR'04*, volume 3170 of *LNCS*, pages 512–528. Springer, 2004. doi:10.1007/978-3-540-28644-8_33.
- 23 T. Wilke. Alternating tree automata, parity games, and modal μ -calculus. *Bull. Belgian Math. Soc.*, 8(2):359–391, 2001. doi:10.36045/bbms/1102714178.

A Proof of Lemma 7

► **Lemma 7.** *Let φ be an \mathcal{L}_μ formula. Then the bottom-up unfolding of φ (cf. Def. 2) and the top-down unfolding of φ (cf. Def. 5) are equivalent to the unfolding defined in Def. 6. In particular, the top-down unfolding is well-defined.*

Proof. We first show that the top-down unfolding of φ is equivalent to the unfolding defined in Def. 6. W.l.o.g. let $\varphi = \sigma_{j+1}X_{j+1}.\psi_{j+1}$ be a formula with fixpoint variables X_1, \dots, X_k such that at most X_1, \dots, X_j for $j \leq k-1$ appear freely in φ . Let $n \geq 0$. We show that

$$(\psi_{j+1}[\psi_1/X_1, \dots, \psi_j/X_j])^{(i, n, \dots, n)} \equiv (\psi_{j+1}^i[\psi_1/X_1, \dots, \psi_j/X_j])^n \quad (1)$$

for closed and fixpoint-free formulas ψ_1, \dots, ψ_j and $i \leq n$. Note that the left term refers to the unfolding according to Def. 6, while the right one refers to the i th unfolding of the single fixpoint X_{j+1} , followed by the n th top-down unfolding of all remaining fixpoint-subformulas. The claim of the Lemma regarding the top-down unfolding procedure then follows with $j = 0$ and $i = n$.

We start with an induction over j . For the base case $j = k-1$ it follows that ψ_j does contain only one fixpoint-subformula and together with the observation that the ψ_1, \dots, ψ_j are closed and fixpoint-free formulas, both sides of Eq. 1 coincide with a single fixpoint unfolding. Let $j < k-1$ and assume that the result has been shown for all j' with $j < j' \leq k-1$ and for all $i \leq n$. We show the result for j by induction on i . For $i = 0$, we have that $\psi_{j+1}^0 = \hat{\sigma}_{j+1}$ and, hence $(\psi_{j+1}^0[\psi_1/X_1, \dots, \psi_j/X_j])^n = \hat{\sigma}_{j+1}$ and $(\psi_{j+1}[\psi_1/X_1, \dots, \psi_j/X_j])^{(0, n, \dots, n)} = \hat{\sigma}_{j+1}$ as well. Next let $i > 0$ and assume that the result has been proven for all $i' \leq i$. We show this case by another induction on the structure of ψ_{j+1} . The base case a , as well as the boolean cases and \bigcirc are straightforward. The interesting cases are the base case of a fixpoint variable $X_{j'}$ for $j' \leq j$, the case of X_{j+1} itself and the case of a fixpoint definition of the

form $\sigma_{j'} X_{j'} \cdot \psi_{j'}$ for $j' > j + 1$. The first case results in the same subformula, because in both versions of the unfolding $\psi_{j'}$ is substituted for $X_{j'}$ before the unfolding, and this is a closed and fixpoint-free formula. For X_{j+1} itself, we use the induction hypothesis of the induction over i , namely that $(X_{j+1}[\psi_1/X_1, \dots, \psi_j/X_j])^{(i, n, \dots, n)} = (\psi_{j+1}[\psi_1/X_1, \dots, \psi_j/X_j])^{(i-1, n, \dots, n)}$ is equivalent to $(\psi_{j+1}^{i-1}[\psi_1/X_1, \dots, \psi_j/X_j])^n$. For the case of a formula of the form $\sigma_{j'} X_{j'} \cdot \psi_{j'}$, we use the induction hypothesis for the induction over j , namely that

$$\begin{aligned} & (\psi_{j'}[\psi_1/X_1, \dots, \psi_j/X_j, \psi'_{j+1}/X_{j+1}, \psi'_{j+2}/X_{j+2}, \dots, \psi'_{j'-1}/X_{j'-1}])^{(n, \dots, n)} \equiv \\ & (\psi_{j'}^n[\psi_1/X_1, \dots, \psi_j/X_j, \psi''_{j+1}/X_{j+1}, \psi''_{j+2}/X_{j+2}, \dots, \psi''_{j'-1}/X_{j'-1}])^n \end{aligned}$$

for $\psi'_{j+1} = (\psi_{j+1}[\psi_1/X_1, \dots, \psi_j/X_j])^{(i-1, n, \dots, n)}$ and $\psi''_{j+1} = (\psi_{j+1}^{i-1}[\psi_1/X_1, \dots, \psi_j/X_j])^n$, which, by the induction hypothesis of the induction over i , are equivalent.¹ This finishes the induction over ψ_{j+1} and with it, the induction over i , and the induction over j .

For the bottom-up unfolding of Def. 2, we show equivalence to the unfolding from Def. 6 by showing that unfolding an innermost formula will not change the formula generated by either procedure. The result then follows by an induction over the ordering of fixpoints used in the bottom-up unfolding. Let φ be a fixpoint formula, and let X_1, \dots, X_k be an enumeration of its fixpoint variables such that $X_i \not\prec_{\text{fp}} X_j$ for $j > i$, and let $\sigma_i X_i \cdot \psi_i$ be the defining formula of X_i . Let $\varphi[\psi_k^n/\sigma_k X_k \cdot \psi_k]$ be the formula obtained by unfolding X_k n times. If we can show that, for all subformulas ψ of φ , and for all $s = (i_1, \dots, i_{k-1}, n)$, we have $\psi^s \equiv (\psi[\psi_k^n/\sigma_k X_k \cdot \psi_k])^s$, we are done. We show this by induction on the lexicographical ordering of the tuple, i.e starting with $s = (0, \dots, 0, n)$, for which the result clearly holds. Assume that we have shown it for all s' that are lexicographically smaller than s . We show the result by another induction on ψ . The base case a , the boolean cases, and the case \bigcirc are straightforward. The interesting cases are a variable X_j for $j < k$, a fixpoint definition of the form $\sigma_j X_j \cdot \psi_j$ for $j < k$, and the fixpoint definition of the form $\sigma_k X_k \cdot \psi_k$. For the case of X_j , if $s(j) = 0$, the result is immediate. For the case that $s(j) > 0$, and for the case of $\sigma_j X_j \cdot \psi_j$, the formula to be substituted is $\text{fp}(X_j)^{s[j--]}$, respectively $(\text{fp}(X_j)[\psi_k^n/\sigma_k X_k \cdot \psi_k])^{s[j--]}$, for which the result holds since $s[j--]$ is lexicographically smaller than s . For the case $\sigma_k X_k \cdot \psi_k$, there are two possibilities. If $s(j) = 0$ for all $0 \leq j < k$, then the result follows from the base case. If not, note that X_k is minimal w.r.t. $\prec_f p$ and, hence, its defining formula $\text{fp}(X_k) = \psi_k$ contains no fixpoint definitions itself. However, it can contain free fixpoint variables from among the X_1, \dots, X_{k-1} . We first note that, by definition, $\text{fp}(X_k)^s$ for $s = (i_1, \dots, i_{k-1}, n)$ is equivalent to $\psi_k^n[\psi_1/X_1, \dots, \psi_{k-1}/X_{k-1}]$, where ψ_j is $\hat{\sigma}_j$ if $s[j] = 0$ and $\text{fp}(X_j)^{s[j--]}$ otherwise. But, using the induction hypothesis for the ψ_i , this is exactly $(\sigma_k X_k \cdot \psi_k[\psi_k^n/\sigma_k X_k \cdot \psi_k])[\psi_1/X_1, \dots, \psi_{k-1}/X_{k-1}]$. This finishes the proof. \blacktriangleleft

B Proof of Lemma 13

► **Lemma 13.** *Let w be an infinite word. If w has finite convergence for the set of all unipolar \mathcal{L}_μ formulas, it has finite convergence for the set of all alternation-free \mathcal{L}_μ formulas.*

Proof. Let φ an alternation-free \mathcal{L}_μ formula. The set of fixpoint variables of φ can be partitioned into the sets $\mathcal{X}_1, \dots, \mathcal{X}_k$ as described in Sec. 2. First, we show by induction on these sets that all fixpoint-subformulas of φ can be unfolded starting from \mathcal{X}_k and ending with \mathcal{X}_1 while preserving equivalence over w . First, note that for all $X \in \mathcal{X}_k$ there is no fixpoint-subformula in $\text{fp}(X)$ that has a different polarity. This implies that all fixpoint-subformulas of

¹ The equality $\psi'_l = \psi''_l$ for $j + 2 \leq l \leq j' - 1$ can also be inferred via induction for a previous j' .

φ with $X \in \mathcal{X}_k$ are unipolar, but not necessarily closed. If we take a fixpoint-subformula with $X \in \mathcal{X}_k$ such that there is no $X' \in \mathcal{X}_k$ with $X' <_{\text{fp}} X$ it follows from the alternation-freeness of φ that $\text{fp}(X)$ is also closed and, thus, by the assumption of the lemma that there is an equivalent unfolding over w . If we replace all such fixpoint-subformulas with their respective equivalent unfolding we have unfolded all fixpoint-subformulas with $X \in \mathcal{X}_k$. Under the assumption that all fixpoint subformulas with $X \in \mathcal{X}_i$ are already unfolded, we can infer with the same arguments that there is an equivalent unfolding for all fixpoint subformulas of \mathcal{X}_{i-1} . By the principle of induction this shows that there is some equivalent finite unfolding of φ over w . What is left to argue is that there is a uniform one, i.e., that there is n such that φ^n is equivalent to φ . Note that for each closed, unipolar fixpoint subformula $\sigma X.\psi$ with equivalent unfolding ψ^m it holds that $\psi^{m'}$ with $m' \geq m$ is equivalent as well. As the number of fixpoint subformulas in φ is finite, n is given by the maximum number of unfoldings needed for any fixpoint subformula. \blacktriangleleft

C Additional Material for Section 6

We give a brief introduction to HFL on words. A more thorough introduction for HFL on arbitrary LTS can be found in e.g. [22]. We also point out that the exposition in Sect. 6 makes use of syntactic sugar that is not directly covered by the pure syntax. The constructs, however, are all straightforward using only principles which are standard in λ -calculi. For instance further below we explain that the subformula $\langle a \rangle$ is used to abbreviate something like $\lambda x.\langle a \rangle x$, so this simply makes use of η -conversion.

Types. Consider the set of types defined via

$$\tau ::= \bullet \mid \tau \rightarrow \tau.$$

The type \bullet is the base type of subsets of \mathbb{N} , for example those defined by an \mathcal{L}_μ formula. A type $\tau_1 \rightarrow \tau_2$ is inhabited by monotone functions from τ_1 to τ_2 .

Such a type induces a lattice over a given word w via the following definition

$$\begin{aligned} \llbracket \bullet \rrbracket^w &= (\mathcal{P}(\mathbb{N}), \subseteq) \\ \llbracket \tau_1 \rightarrow \tau_2 \rrbracket^w &= (\llbracket \tau_2 \rrbracket^w \rightarrow \llbracket \tau_1 \rrbracket^w, \sqsubseteq_{\tau_1 \rightarrow \tau_2}) \end{aligned}$$

where $\llbracket \tau_2 \rrbracket^w \rightarrow \llbracket \tau_1 \rrbracket^w$ denotes the set of functions from $\llbracket \tau_1 \rrbracket^w$ to $\llbracket \tau_2 \rrbracket^w$ and $\sqsubseteq_{\tau_1 \rightarrow \tau_2}$ is defined as the pointwise order via $f \sqsubseteq_{\tau_1 \rightarrow \tau_2} g$ iff $f(x) \sqsubseteq_{\tau_2} g(x)$ for all $x \in \llbracket \tau_1 \rrbracket^w$. Here, \sqsubseteq_\bullet ordinary set inclusion \subseteq . All these lattices are complete since $\llbracket \bullet \rrbracket^w$ is complete, and a lattice of functions is complete if the functions are into a complete lattice and ordered pointwise.

Syntax. Let $\mathcal{V} = \{x, y, \dots\}$ be a set of λ variables. The syntax of HFL extends that of \mathcal{L}_μ to

$$\varphi ::= a \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid X \mid x \mid \mu(X : \tau).\varphi \mid \nu(X : \tau).\varphi \mid \lambda(x : \tau).\varphi \mid \varphi \varphi$$

where τ is a type.

The intuition behind the operators that are new in comparison to \mathcal{L}_μ is as follows.

- $\lambda x.\varphi$ defines an anonymous function that consumes an argument, bound to x , and returns the value of φ if x is set to that argument.
- $\varphi \psi$ denotes the application of ψ to φ , and
- the fixpoints can now be of higher order, i.e. define function type objects.

$$\begin{array}{c}
 \frac{}{\Gamma \vdash a : \bullet} \quad \frac{}{\Gamma, X : \tau \vdash X : \tau} \quad \frac{}{\Gamma, x : \tau \vdash x : \tau} \quad \frac{\Gamma \vdash \varphi_1 : \bullet \quad \Gamma \vdash \varphi_2 : \bullet}{\Gamma \vdash \varphi_1 \vee \varphi_2 : \bullet} \\
 \\
 \frac{\Gamma \vdash \varphi_1 : \bullet \quad \Gamma \vdash \varphi_2 : \bullet}{\Gamma \vdash \varphi_1 \wedge \varphi_2 : \bullet} \quad \frac{\Gamma \vdash \varphi : \bullet}{\Gamma \vdash \bigcirc \varphi : \bullet} \quad \frac{\Gamma, X : \tau \vdash \varphi : \tau'}{\Gamma \vdash \lambda(X : \tau). \varphi : \tau \rightarrow \tau'} \quad \frac{\Gamma, X : \tau \vdash \varphi : \tau}{\Gamma \vdash \mu(X : \tau). \varphi : \tau} \\
 \\
 \frac{\Gamma, X : \tau \vdash \varphi : \tau}{\Gamma \vdash \nu(X : \tau). \varphi : \tau} \quad \frac{\Gamma \vdash \varphi : \tau \rightarrow \tau' \quad \Gamma \vdash \varphi' : \tau}{\Gamma \vdash \varphi \varphi' : \tau'}
 \end{array}$$

■ **Figure 1** The type system of HFL.

Note that we have chosen not to introduce negation. HFL admits negation normal form [18], whence negation only needs to occur on front of atomic propositions. Over words, the formula $\neg a$ however is equivalent to $\bigvee_{b \in \Sigma, b \neq a} b$ as stated for \mathcal{L}_μ already, so negation can be avoided altogether.

An advantage of this avoidance of negation is the simplification of the type system to monotone functions only. The original definition of HFL includes negation as a syntactic construct at the expense of a slightly more complex type systems which needs to keep track of antitonicity information so that fixpoints are guaranteed to exist due to monotonicity.

Without this, the only purpose of the type system is to avoid misapplications as in $a b$ for instance which cannot be given proper meaning. Another example is $a \vee \lambda(x : \mathbb{N}). x \wedge b$.

In the absence of negation, an HFL formula φ is said to be well-typed if the statement $\emptyset \vdash \varphi$ can be derived via the rules shown in Fig. 1. The sequence Γ on the left of a typing statement is called a typing context and collects typing hypotheses.

Semantics. In order to endow well-typed HFL formulas with semantics, we extend environments to \mathcal{V} , i.e. environments can also store values for λ -variables which may be higher-order objects depending on the type of the variable. The semantics of an HFL formula φ is given inductively as per the following:

$$\begin{aligned}
 \llbracket a \rrbracket_\eta^w &= \{i \in \mathbb{N} \mid w[i] = a\} \\
 \llbracket X \rrbracket_\eta^w &= \eta(X) \\
 \llbracket x \rrbracket_\eta^w &= \eta(x) \\
 \llbracket \varphi \vee \psi \rrbracket_\eta^w &= \llbracket \varphi \rrbracket_\eta^w \cup \llbracket \psi \rrbracket_\eta^w \\
 \llbracket \varphi \wedge \psi \rrbracket_\eta^w &= \llbracket \varphi \rrbracket_\eta^w \cap \llbracket \psi \rrbracket_\eta^w \\
 \llbracket \bigcirc \varphi \rrbracket_\eta^w &= \{i \in \mathbb{N} \mid i + 1 \in \llbracket \varphi \rrbracket_\eta^w\} \\
 \llbracket \lambda(x : \tau). \varphi \rrbracket_\eta^w &= f \in \llbracket \tau \rightarrow \tau' \rrbracket^w \quad \text{where f.a. } d \in \llbracket \tau \rrbracket^w . f(d) = \llbracket \varphi \rrbracket_{\eta[x \mapsto d]}^w \\
 &\quad \text{with } \tau' \text{ the type of } \varphi \\
 \llbracket \varphi \psi \rrbracket_\eta^w &= \llbracket \varphi \rrbracket_\eta^w (\llbracket \psi \rrbracket_\eta^w) \\
 \llbracket \mu(X : \tau). \varphi \rrbracket_\eta^w &= \bigsqcap \{d \subseteq \llbracket \tau \rrbracket^w \mid \llbracket \varphi \rrbracket_{\eta[X \mapsto d]}^w \sqsubseteq_\tau d\} \\
 \llbracket \nu(X : \tau). \varphi \rrbracket_\eta^w &= \bigsqcup \{d \subseteq \llbracket \tau \rrbracket^w \mid d \sqsubseteq_\tau \llbracket \varphi \rrbracket_{\eta[X \mapsto d]}^w\}
 \end{aligned}$$

The formula φ from Sect. 6. We give some additional explanation about the formula φ defined in Sec. 6. Recall that

$$\varphi = (\nu(X : \tau). \lambda(f, g : \mathbb{N} \rightarrow \mathbb{N}). f(\mathbf{tt}) \wedge X(f^2 \circ g \circ f^2, g^2 \circ f \circ g^2)) (\langle b \rangle, \langle a \rangle).$$

Notation such as $\lambda(f, g : \mathbb{N} \rightarrow \mathbb{N}). \dots$ can easily be seen to be an abbreviation of the longer $\lambda(f : \dots). \lambda(g : \dots). \dots$, and the same goes for the application to X written in a similar style. Clearly, neither function composition nor the function $\langle a \rangle$ are in the official syntax of HFL. The function $\langle a \rangle$ can be written in standard syntax as $\lambda(x : \mathbb{N}). a \wedge \bigcirc x$. Function composition, here between five functions, is an abbreviation for $\lambda(x : \mathbb{N}). f(f(g(f(f(x))))))$, respectively $\lambda(x : \mathbb{N}). g(g(f(g(g(x))))))$.

The intuition given for the semantics of φ is already close to the true semantics of φ . Since the Kleene Fixpoint Theorem applies in this setting, too, we can use it to obtain the semantics of φ on w_∞ . As in Sec. 6, write ψ for the defining formula of $\lambda f, g. f(\mathbf{tt}) \wedge X(f^2 \circ g \circ f^2, g^2 \circ f \circ g^2)$, write ψ^0 for $\lambda f, g. \mathbf{tt}$ and ψ^{i+1} for $\psi[\psi^i/X]$. Given some functions f and g , we introduce the following functions:

$$\begin{aligned} f_1(x) &= f(x) & f_{i+1} &= f_i(f_i(g_i(g_i(g_i(x)))))) \\ g_1(x) &= g(x) & g_{i+1} &= g_i(g_i(f_i(g_i(g_i(x)))))) \end{aligned}$$

Then ψ^i defines the function $f, g \mapsto f_1(\mathbf{tt}) \wedge f_2(\mathbf{tt}) \wedge \dots \wedge f_i(\mathbf{tt})$ by induction on i . Hence, the semantics of X is $f, g \mapsto \bigwedge_{i \in \mathbb{N}} f_i(\mathbf{tt})$. Applied to the arguments $\langle a \rangle$ and $\langle b \rangle$, this yields, by abuse of notation, $\bigwedge_{i \in \mathbb{N}} \langle \alpha_i \rangle \mathbf{tt}$, which is as claimed in Sec 6. The remarks on the infinite convergence process of φ on w_∞ then follow.



Dots & Boxes Is PSPACE-Complete

Kevin Buchin  

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Mart Hagedoorn 

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Irina Kostitsyna  

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Max van Mulken 

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

Exactly 20 years ago at MFCS, Demaine posed the open problem whether the game of Dots & Boxes is PSPACE-complete. Dots & Boxes has been studied extensively, with for instance a chapter in Berlekamp et al. *Winning Ways for Your Mathematical Plays*, a whole book on the game *The Dots and Boxes Game: Sophisticated Child's Play* by Berlekamp, and numerous articles in the *Games of No Chance* series. While known to be NP-hard, the question of its complexity remained open. We resolve this question, proving that the game is PSPACE-complete by a reduction from a game played on propositional formulas.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Dots & Boxes, PSPACE-complete, combinatorial game

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.25

1 Introduction

Dots & Boxes is a popular paper-and-pencil game that is played by two players on a grid of dots. The players take turns connecting two adjacent dots. If a player completes the fourth side of a unit box, the player is awarded a point and an additional turn. When no more moves can be made, the player with the highest score wins the game.¹

Originally described in 1883 [29], Dots & Boxes has since received a considerable amount of attention in the research community. In *Winning Ways for Your Mathematical Plays*, Berlekamp, Conway, and Guy [6] were among the first to discuss a number of interesting mathematical properties of the game. Later, Berlekamp [5] wrote an entire book *The Dots-and-Boxes game: Sophisticated Child's Play* about the game, in particular discussing winning strategies in particular positions. Since then, the mathematics of Dots & Boxes and variants have been discussed in many papers and books [1, 2, 7, 12, 16, 21, 22, 26, 30, 31, 33, 34]. There is also a rich body of work on solvers for Dots & Boxes [3, 4, 11, 27, 35].

Berlekamp et al. [6] argue that deciding the winner of a generalized version of Dots & Boxes, called *Strings-and-Coins*, is NP-hard. In this game, players take turns in removing edges of a given graph, scoring a point when they isolate a vertex. When restricted to the dual graph of a square grid, this corresponds to a dual formulation of Dots & Boxes. Eppstein [17] notes that the reduction given by Berlekamp et al. should extend to Dots & Boxes, and a formal proof of the NP-hardness is given in [8].

Exactly 20 years ago at MFCS, Demaine posed the open problem whether Dots & Boxes is PSPACE-complete [13]. Bounded two-player games, like Dots & Boxes, (that is, games in which the number of moves is bounded) naturally lie in PSPACE, since a Turing machine using space polynomial in the board size is able to search the entirety of the game

¹ For a visual explanation of the game see <https://youtu.be/KboGyIi1P6k>, last accessed 6.5.2021



space. PSPACE-hardness of many bounded two-player games is shown by a reduction from *Generalized Geography*, which is proven PSPACE-complete by Lichtenstein and Sipser [28]. For example, the PSPACE-completeness of Reversi [24], uncooperative UNO [14], and Tic-Tac-Toe [23] were shown by a reduction from Generalized Geography. However, unlike Dots & Boxes, the setting of Generalized Geography prescribes a stricter order on players' moves, making a reduction to Dots & Boxes challenging to obtain.

In their seminal work, Hearn and Demaine [20, 21] introduce *Constraint Logic*, a framework for analyzing complexity of games and puzzles. Inspired by Flake and Baum's proof of Rush Hour [18], it specifies a type of game played on a constraint graph. The framework includes bounded/unbounded state spaces and single/two-player variations. In the same work, Hearn and Demaine go on to provide a number of simpler reductions for various known PSPACE-complete games (including Rush Hour), as well as new proofs for several PSPACE-complete games. However, the Constraint Logic framework is intended for proving hardness of partisan games (games in which the moves available to the two players are different), whereas Dots & Boxes is not a partisan game.

Strings-and-Coins and the related game of *Nimstring* were very recently (while we were preparing this submission) proven to be PSPACE-complete by Demaine and Diomidov [15] by a reduction from a game on a DNF formula $G_{pos}(\text{POS DNF})$ [32]. But, as they point out, their results do not apply to Dots & Boxes, since the game positions they construct rely on multi-graphs (which additionally are neither planar nor have a maximum degree of 4). Specifically, they propagate signals through multi-edges consisting of a polynomial number of parallel edges, and the winner is the player who removes the last edge. As consequence, our reduction bears little commonalities with theirs.

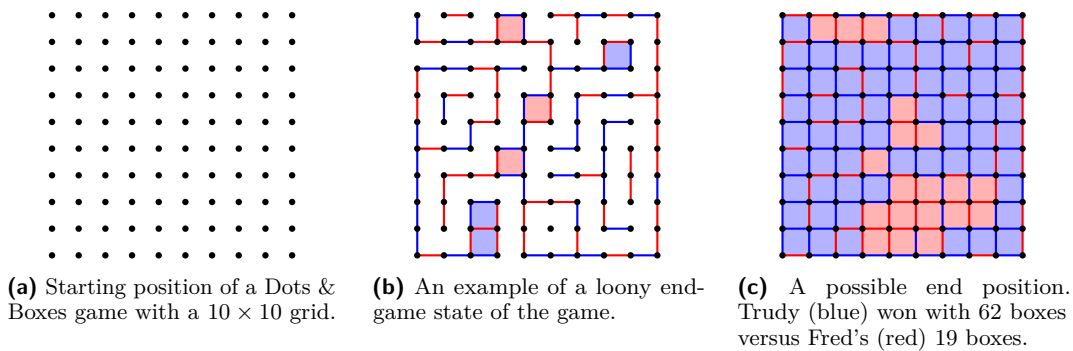
In this paper, we prove that Dots & Boxes is PSPACE-complete by a reduction from $G_{pos}(\text{POS CNF})$. The starting point of our construction are strategies for Dots & Boxes endgames that were also used to prove NP-hardness. However, the NP-hardness is proven by having one player be in control, such that there is only one way for the other player to respond. This de facto makes the game into a 1-player game. For PSPACE-hardness we need both players to have choices, making it a true 2-player game. This gives a lot of freedom to the players, and makes it much more difficult to construct gadgets to control the gameplay, in particular because moves and scoring opportunities for one player – if not played immediately – are also available to the other player.

In Section 1.1 we discuss the gameplay of Dots & Boxes, and introduce terminology coined by Berlekamp et al. [6]. In Section 2 we present the general structure of our reduction, and then describe our gadgets in Section 3. In Section 4 we first show properties of optimal play for both players and finally prove PSPACE-hardness.

1.1 Dots & Boxes

On the surface, Dots & Boxes is quite a simple game. The starting and a typical final position for a 10×10 grid are shown in Figure 1. We refer to the players playing the blue and the red colors as Trudy and Fred, respectively. The color of a line connecting two dots indicates which player drew it, and the color of a box indicates which player closed it.

Consider a dual graph G of a board of Dots & Boxes, where a node in G corresponds to a box or the unbounded face, and a pair of nodes in G is connected with an edge if the corresponding move is still available, i.e., the line between the boxes has not been drawn. Let the *degree* of a box be the degree of the corresponding node in G .

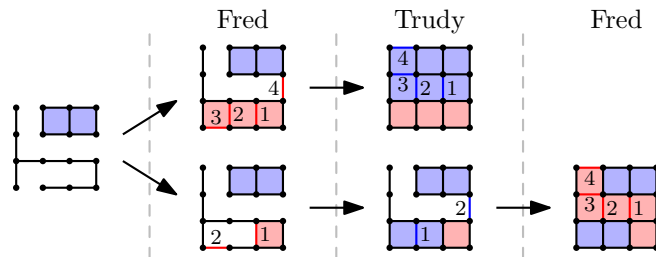


■ **Figure 1** Typical starting, intermediate, and final position of a Dots & Boxes game.

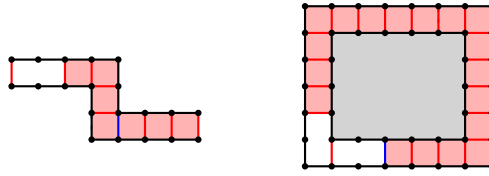
In Dots & Boxes, a typical game usually results in a board state that consists exclusively of moves that open the possibility for the opponent to claim a number of boxes in their next turn (see Figure 1b). That is, in this state there are no degree-1 boxes, but any move made by a player creates a degree-1 box that can be immediately claimed by the opponent. Consider such a board configuration S and any available move ℓ in it. At least one box b incident to ℓ has degree two in S (before the move ℓ is made). Consider a maximal component σ of degree-2 boxes in S containing b . There are two cases, either σ is a *chain* ending in boxes of degree higher than two (or the outer face), or σ is a *cycle*. Then we say that a player making the move ℓ *opens* the chain (cycle) σ for the opponent.

To devise a good strategy for Dots & Boxes, it is important to note that a player is not obliged to claim a box whenever they can. It is sometimes beneficial for a player to sacrifice a small number of boxes for long-term gain. Consider the position in Figure 2, and let it be Fred's (red) turn. Here, Fred could claim the bottom three boxes (Figure 2 (top)). However, after doing so Fred has to make an extra move, allowing Trudy (blue) to claim the remaining four boxes and win the game. But by sacrificing two boxes (Figure 2 (bottom)), Fred can force Trudy to make another move and open the middle chain for him to claim. That way, Fred loses two boxes in the bottom chain, but gains all four boxes in the middle chain.

In *Winning Ways*, Berlekamp et al. [6] refer to the moves sacrificing a small number of boxes but passing the turn onto the opponent as *double-dealing* moves. Double-dealing moves can be made in chains of boxes, sacrificing two boxes, and in cycles, sacrificing four boxes (see Figure 3). Each double-dealing move is usually immediately followed by the opponent making at least one *double-cross* move, i.e., a move that closes two boxes at once. Note that



■ **Figure 2** Two possible plays that Fred (red) can do. Fred can choose to claim all the available boxes (top) and lose the game, or to perform a double-dealing move sacrificing two boxes (bottom, second state, edge 2), and win the game. The order of the edges that are played by Trudy or Fred in one turn is indicated by edge labels. This example is borrowed from *Winning Ways*, chapter 16 [6].



■ **Figure 3** Double-dealing move by Fred (red). If Trudy (blue) opens a chain (or a cycle), Fred can claim a sequence of boxes. To pass the turn back to Trudy, Fred can leave two (or four) boxes unclaimed.

double-dealing moves are only possible in *long* chains of at least three boxes, and in cycles. (Chains of length one do not have enough boxes for a double-dealing move, and a chain of length two can be opened by selecting the middle edge, thus preventing the opponent from playing a double-dealing move.)

Berlekamp et al. [6] refer to moves opening a long chain or a cycle as *loony* moves. Making a loony move is not always a choice, since at some point in the game, all unclaimed boxes are part of long chains and cycles as in Figure 1b. Such positions are referred to as a *loony endgame*. Note that in chains of length ≥ 4 and cycles of length ≥ 8 , the player making the double-dealing moves scores at least as many boxes as their opponent. Thus, in loony endgames with chains of length ≥ 4 and cycles of length ≥ 8 , under optimal play, the game consists of one player making loony moves (opening chains and cycles), and the other player claiming all but two or four boxes, and making double-dealing moves to pass the turn back to the opponent [6]. Here, the player making the double-dealing moves is always better off, since each chain or cycle yields at least as many boxes to this player as it yields to their opponent. This player is thus referred to as being *in control* of the game. The benefit of being in control can be seen in Figure 1c, which is the end result of Trudy being in control of the loony endgame shown in Figure 1b.

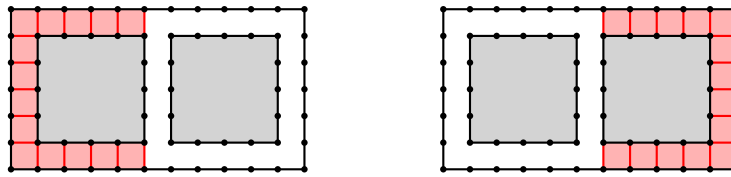
In *Winning Ways*, Berlekamp et al. [6] argue that the player not in control has to maximize the number of disjoint cycles to maximize their score, because double-dealing moves in cycles yield twice as many boxes as double-dealing moves in chains. Since this property is important for our reduction, we restate it here and, for completeness, present the argument in the appendix.

► **Lemma 1.** *Let the configuration of a loony endgame contain k boxes with degree higher than 2, let T be the sum of the degrees of these boxes, and let c be the maximum number of disjoint cycles in the configuration. Then, the player who is not in control can claim at most $4c + T - 2k - 4$ boxes.*

Note that in this lemma we only count boxes; even though the outer face contributes to the degrees of adjacent boxes, it is not counted here.

2 Structure of the construction

To show that Dots & Boxes is PSPACE-hard we reduce from the game $G_{pos}(\text{POS CNF})$, introduced and proven PSPACE-complete by Schaefer [32]. The game is played by two players, Trudy and Fred, on a positive CNF formula \mathcal{F} . The players take turns picking a variable that has not yet been chosen. Variables picked by Trudy are set to TRUE, variables picked by Fred are set to FALSE. When all variables have been chosen, the game ends. Trudy wins if formula \mathcal{F} evaluates to TRUE, and Fred wins if formula \mathcal{F} evaluates to FALSE.



■ **Figure 4** A choice of a cycle can encode the value of a signal.

Given a positive CNF formula \mathcal{F} with n variables and m clauses, we construct an instance of Dots & Boxes in which Trudy has a winning strategy if and only if she also has a winning strategy in the corresponding instance of $G_{pos}(\text{POS CNF})$. For simplicity we assume that n is even, so that Trudy and Fred get to assign values to the same number of variables. If the number of variables in \mathcal{F} is odd, we can introduce dummy variables without changing the outcome of a game such that the total number of the variables becomes even. For each variable and clause of \mathcal{F} we construct a **variable** and a **clause** gadget, respectively. We place the **variable** gadgets in a row at the top of the board of Dots & Boxes, and the **clause** gadgets in a row at the bottom. We connect the **variable** gadgets to their corresponding **clause** gadgets using the **wire** gadgets, which transfer the values of the variables to the clauses. If a clause consists of more than one variable, the wires from these variables must pass through an **or** gadget. Since the signals propagating from the variables may need to cross each other, we construct a **crossover** gadget that preserves the values in the two crossing wires. In our instance of Dots & Boxes, only the gadgets contain available moves. The remaining boxes on the board have all the incident edges present.

As we detail in Section 4, after the values of the variables are set, the game enters a loony endgame where Fred is in control. Then Trudy's winning strategy reduces to selecting a maximum set of disjoint cycles \mathcal{C}_{\max} in the remaining configuration (Lemma 1). In the remainder of this paper, we describe a strategy for both players referred to as *regular play*. Later, we will show that following the regular strategies is optimal for both players, in the sense that it maximizes their scores. Under regular play, Trudy opens all the chains outside of \mathcal{C}_{\max} first, gaining two boxes per chain, and opens the chosen cycles last, gaining four boxes per cycle except the last one. Regular play for Fred is to ensure that he will be in control when the loony endgame starts. After entering the loony endgame, regular play for Fred is simply making double-dealing moves until his very last turn.

Most of our gadgets consist of partially overlapping cycles of boxes. The choice of a set of disjoint cycles determines the value of a signal. For example, in Figure 4 the choice of the left vs. right cycle can encode the value TRUE vs. FALSE. Of course, Trudy could join the cycles together to select the outermost cycle, but this, as we show later, will not be more beneficial.

As both players must have a choice in picking which variable to set, the instance of Dots & Boxes cannot yet be in a loony endgame. Thus, the **variable** gadgets, which we describe in Section 3.4, contain non-loony moves instrumental in setting the value of a variable. We ensure that under regular play the variables are set in alternating fashion, where Trudy sets them to TRUE, and Fred sets them to FALSE. Once all variables are set, the loony endgame is entered. At this point Fred is in control of the game, and it is up to Trudy to maximize her score by maximizing the number of disjoint cycles in \mathcal{C}_{\max} . The regular play by Trudy results in a correct propagation of the signals from the variables to the clauses.

To ensure that regular play by both players in the instance of Dots & Boxes corresponds to a valid $G_{pos}(\text{POS CNF})$ game, our gadgets need to give a specific number of boxes to

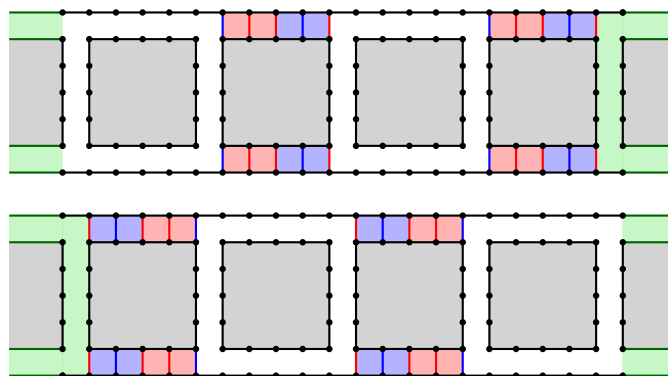
Trudy depending on the signal values. We will show that after the variable values have been set, under regular play, Trudy can maximize her score only if the signals are propagated correctly. Every gadget, except for the clause, yields the same number of disjoint cycles independent of the signals passing through the gadget. Only the clause gadget gives more cycles to Trudy if a TRUE signal reaches it. Exactly half of the variables are set to TRUE, and half to FALSE. Thus we can tune the starting score count such that the game is won by Trudy if and only if all the clauses are satisfied.

3 Gadgets

In this section we provide the details of the gadgets used in our reduction. When describing the gadgets below, for a simpler exposition, we assume that the moves that Trudy and Fred make follow the following sequence. First, in the first n moves Trudy and Fred set all the variables to TRUE and FALSE respectively. Afterwards, when the loony endgame is entered, the order in which Trudy selects which cycles to add to the disjoint set of cycles \mathcal{C}_{\max} is from the top to bottom, that is, from the variables, through the outgoing wires, through the crossover and or gadgets, and finally down to the clause gadgets. Later, in Lemma 7, we will show that, indeed, under regular play Trudy and Fred start by setting all the variables. Furthermore, we will argue that the outcome of the game depends only on the choice of the cycles in \mathcal{C}_{\max} , and not on the order in which Trudy selects them.

3.1 Basic wiring

Signals from the variable gadgets are propagated to the clause gadgets through wires. A wire gadget consists of a chain of an even number of partially overlapping cycles (see Figure 5). The first cycle in the wire overlaps with the gadget from which the signal is propagated, and the last cycle overlaps with the gadget towards which the signal is propagated. Consider some wire w , let C_i be its first cycle overlapping with gadget G_i , and let C_o be its last cycle overlapping with gadget G_o . If C_i is disjoint from the cycles of G_i that Trudy adds to \mathcal{C}_{\max} , then we say that the input signal to the wire is TRUE; otherwise, if C_i overlaps with one of the cycles of G_i in \mathcal{C}_{\max} , the input value is FALSE. If Trudy does not add C_o to \mathcal{C}_{\max} , then the output signal is TRUE, and the output signal is FALSE otherwise.



■ **Figure 5** A wire gadget consisting of four overlapping cycles and two ways of selecting disjoint cycles. Shown in green are the connections to the adjacent gadgets. Selecting odd cycles in \mathcal{C}_{\max} corresponds to TRUE (top), and selecting even cycles corresponds to FALSE (bottom).

To ensure that Fred always follows the strategy of double-dealing moves, we require that each maximal chain of degree-2 boxes in a wire gadgets contain at least four boxes. That way, Fred receives at least as many boxes in each chain (and cycle) as Trudy, and thus for Fred being in control is always beneficial [6].

Note that, besides the lower bound on the length of a chain, the size and the embedding of the overlapping cycles in a wire can be chosen freely. Thus wires are very flexible in connecting components together, which facilitates the construction.

► **Lemma 2.** *Let a wire w consist of $2k$ partially overlapping cycles. Then, under regular play, if the signal in w changes from FALSE to TRUE, then Trudy can select at most $k - 1$ disjoint cycles from w to add to \mathcal{C}_{\max} . Otherwise, under regular play, Trudy can select k disjoint cycles from w to add to \mathcal{C}_{\max} .*

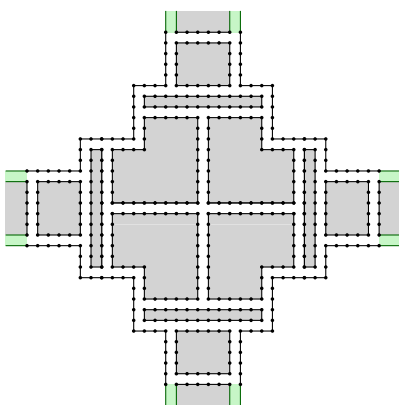
Proof. As we show in Lemma 7, after the first n moves, which Trudy and Fred make in the variable gadgets, the game enters a loony endgame with Fred in control. If the output signal in the wire matches the input signal, then only one of C_i or C_o of w are in \mathcal{C}_{\max} . Then Trudy can select all odd (if $C_i \in \mathcal{C}_{\max}$) or all even (if $C_o \in \mathcal{C}_{\max}$) cycles to add to \mathcal{C}_{\max} , which results in k disjoint cycles. If the the input signal is TRUE, and the output signal is FALSE, then both C_i and C_o are in \mathcal{C}_{\max} . Then Trudy can, for example, select $k - 1$ odd cycles and C_o to add to \mathcal{C}_{\max} , which again results in k cycles in total.

If, however, the input signal is FALSE, and the output signal is TRUE, then neither C_i nor C_o can be in \mathcal{C}_{\max} . This leaves a chain of $2k - 2$ cycles, of which at most $k - 1$ disjoint cycles can be selected to be added to \mathcal{C}_{\max} . ◀

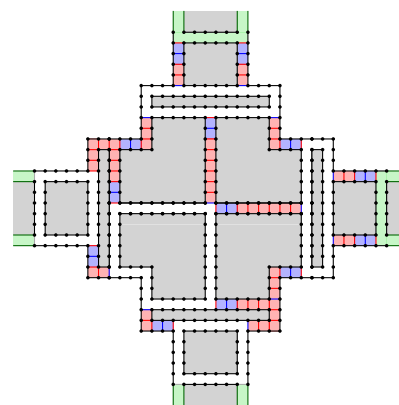
In our construction we ensure that Trudy can win only if she gets k disjoint cycles from a wire, and thus under regular play she cannot flip a signal propagating from a variable from FALSE to TRUE. Flipping a signal from TRUE to FALSE is not beneficial for Trudy, as her goal is to satisfy all the clauses. Nevertheless, flipping a signal from TRUE to FALSE leads to the same number of boxes for her (at least locally within a wire), and is thus allowed.

3.2 Crossover gadget

Since the graph representing G_{pos} (POS CNF) is not necessarily planar, wires may need to cross each other in our construction. We describe a **crossover gadget** that allows two signals to cross while preserving the signal values. The gadget has two inputs and two outputs



■ **Figure 6** The crossover gadget. Connections to the adjacent wires are shown in green.



■ **Figure 7** A possible choice of a set of disjoint cycles. The selection has fourth degree rotational symmetry.

on the opposite sides of the gadget. Let $C_{1,i}$ and $C_{2,i}$ be the input cycles of the gadget, and $C_{1,o}$ and $C_{2,o}$ be the output cycles (see Figure 6). An input cycle $C_{*,i}$ is in \mathcal{C}_{\max} if the corresponding input signal is TRUE, and otherwise it is FALSE. An output cycle $C_{*,o}$ is not in \mathcal{C}_{\max} if the output signal is TRUE, and otherwise it is FALSE.

There are four pairwise overlapping cycles $C_a, C_b, C_c,$ and C_d in the middle of the gadget, forming a cross shape. Only one of these cycles can be added to \mathcal{C}_{\max} . A choice of which of these cycles is added to \mathcal{C}_{\max} is in one-to-one correspondence to the input signal values (see Figure 7).

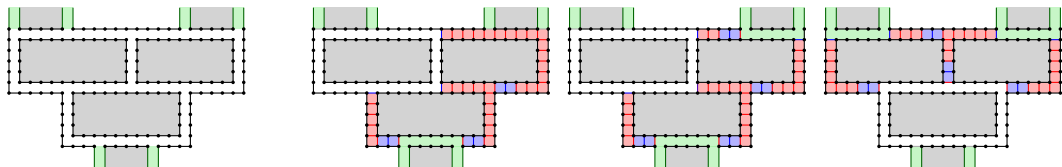
► **Lemma 3.** *Under regular play, if a signal in a crossover gadget changes from FALSE to TRUE, then Trudy can select at most 4 disjoint cycles from the gadget to add to \mathcal{C}_{\max} . Otherwise, under regular play, Trudy can select 5 disjoint cycles from the gadget.*

Proof. If the output signals in the crossover gadget match the input signals, then only one of each pair $\{C_{1,i}, C_{1,o}\}$ and $\{C_{2,i}, C_{2,o}\}$ are in \mathcal{C}_{\max} . Since the four center cycles $C_a, C_b, C_c,$ and C_d all share a single square, only one of these four cycles can be chosen. Then Trudy can select a corresponding cycle from the middle of the gadget, and two more cycles from each signal. For example, a selection of five disjoint cycles for the case when the first input signal is FALSE and the second is TRUE is shown in Figure 7. If an input signal is TRUE, and the corresponding output signal is FALSE, then both $C_{*,i}$ and $C_{*,o}$ are in \mathcal{C}_{\max} . Then Trudy can, for example, make exactly the same choice as in the case where the output signal would have been TRUE.

Assume now, w.l.o.g., that the signal corresponding to $C_{1,i}$ and $C_{1,o}$ changes from FALSE to TRUE in the gadget. That is, neither $C_{1,i}$ nor $C_{1,o}$ are in \mathcal{C}_{\max} . Let C' and C'' be the cycles in the gadget adjacent to $C_{1,i}$ and $C_{1,o}$ respectively. Thus, among cycles $C', C'', C_a, C_b, C_c,$ and C_d at most two cycles can be in \mathcal{C}_{\max} , and therefore at most four cycles can be chosen to be in \mathcal{C}_{\max} . ◀

3.3 Or gadget

The or gadget consists of three pairwise overlapping cycles (see Figure 8 (left)). Two of the cycles partially overlap with an end cycle of an input wire, and one cycle partially overlaps with the output cycle. Let $C_{1,w}$ and $C_{2,w}$ be the last cycles of the two input wire gadgets, and let $C_{1,i}$ and $C_{2,i}$ be the cycles of an or gadget adjacent to these two wires respectively. Let C_o be the third cycle of the or gadget, which is adjacent to an output wire. Cycles $C_{1,w}$ and $C_{2,w}$ are not in \mathcal{C}_{\max} if the input from their corresponding wire is TRUE, and are in \mathcal{C}_{\max} if their input is FALSE. If C_o is not in \mathcal{C}_{\max} then the output of the or gadget is TRUE, and if it is in \mathcal{C}_{\max} then the output value is FALSE. Only one of the three cycles in the or gadget can be selected to be added to \mathcal{C}_{\max} , and thus the output of the gadget can be TRUE only if one of $C_{1,w}$ or $C_{2,w}$ is in \mathcal{C}_{\max} .



■ **Figure 8** The or gadget (left) and the three possible combinations of the input values (right). From left to right: two TRUE inputs, one TRUE and one FALSE input, and two FALSE inputs. The boxes highlighted in green belong to a cycle in an adjacent wire gadget.

► **Lemma 4.** *Under regular play, if both input signals in an or gadget are FALSE but the output signal is TRUE, then Trudy cannot add any cycles from the gadget to \mathcal{C}_{\max} . Otherwise, under regular play, Trudy can select 1 cycle from the gadget to add to \mathcal{C}_{\max} .*

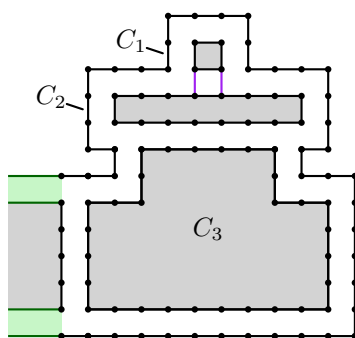
Proof. First consider the case when one of the input signals in the or gadget is TRUE. W.l.o.g., let the signal from the first wire be TRUE, that is $C_{1,w}$ is not in \mathcal{C}_{\max} . Then Trudy can select $C_{1,i}$ to add to \mathcal{C}_{\max} and thus the output from the or gadget would correspond to TRUE. Trudy may as well choose C_o to add to \mathcal{C}_{\max} and make the output of the gadget to be FALSE. In either case, one cycle from the gadget is in \mathcal{C}_{\max} .

If both input signals are FALSE, then both cycles $C_{1,w}$ and $C_{2,w}$ are in \mathcal{C}_{\max} . Thus none of $C_{1,i}$ and $C_{2,i}$ can be in \mathcal{C}_{\max} . Trudy can choose to add C_o to \mathcal{C}_{\max} and make the output of the gadget to be FALSE. If, however, Trudy chooses to make the output of the or gadget TRUE, then C_o is not in \mathcal{C}_{\max} , and thus Trudy cannot select a single cycle to add to \mathcal{C}_{\max} from this or gadget. ◀

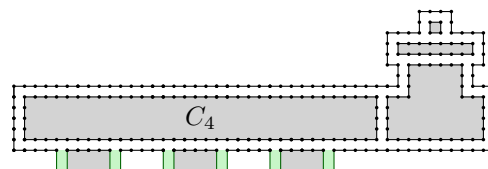
3.4 Variable gadget

The variable gadget is responsible for the assignment of TRUE and FALSE values to the variables of the G_{pos} (POS CNF) instance. It consists of two components: the *value-setting component* (see Figure 9) designed to set the value of the variable, and the *fan-out component* designed to duplicate the variable signal. The whole construction is presented in Figure 10. Let C_1 , C_2 , and C_3 be the three cycles in the value-setting component. The variable gadget is the only gadget that contains non-loony moves; there are two non-loony moves (shown in purple in the figure) at the intersection of C_1 and C_2 .

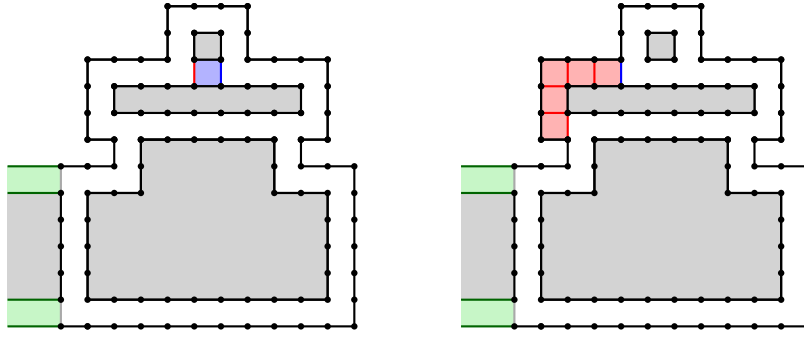
Regular play by both Trudy and Fred is to set all the variables in the first n moves, such that Fred always sets a variable to FALSE and Trudy – to TRUE. Figure 11 shows the two possible value assignments of the variable gadgets. To set a variable to FALSE, Fred plays one of the non-loony moves in the corresponding variable gadget. Then Trudy responds by claiming the one box available (see Figure 11 (left)). This results in the cycles C_1 and C_2 getting merged. To set a variable to TRUE, Trudy opens a side chain of C_2 (see Figure 11 (right)). Then Fred responds by claiming every box in the opened chain, and proceeds to setting the next variable. Note that after Trudy’s move the non-loony moves in the gadget become loony moves (as they are now a part of a long chain).



■ **Figure 9** The value-setting component of the variable gadget. There are two non-loony moves (purple) available, of which only one can be played as a non-loony move.



■ **Figure 10** The complete variable gadget consisting of the value-setting component and the fan-out component. Outgoing wires are shown in green.



■ **Figure 11** The variable is set to FALSE (left) and TRUE (right).

We make two observations which will be useful when proving correctness of the construction and the properties of the regular play in Section 4. First, observe that the non-loony moves come in pairs, one in each variable, such that, for each pair, either both moves in the pair are still non-loony or neither is anymore. We refer to them as *non-loony pairs*. Second, note that in the process of assigning values to the variable gadgets, Trudy gets a box for each variable set to FALSE by Fred, and zero boxes for each variable set to TRUE by herself.

Once the value of a variable is set, it propagates to the outgoing wires through the fan-out component of the variable gadget. The fan-out component simply consists of one cycle C_4 overlapping with the cycle C_3 (see Figure 10), to which multiple wires can be attached. After the variable is set, Trudy can add at most two cycles from it to \mathcal{C}_{\max} . Then, if the variable is set to FALSE, cycle C_4 has to be one of the two selected cycles, and thus the signal propagated into the wires is FALSE. If the variable is set to TRUE, Trudy can add C_1 and C_3 to \mathcal{C}_{\max} , and thus propagate the TRUE value into the wires. By considering the various cases under regular play, we obtain the following lemma.

► **Lemma 5.** *Under regular play, after a variable gadget is assigned a value, if it is set to FALSE but the output signal is TRUE, then Trudy can add at most 1 cycle from the gadget to \mathcal{C}_{\max} . Otherwise, under regular play, Trudy can add 2 cycles from the gadget to \mathcal{C}_{\max} .*

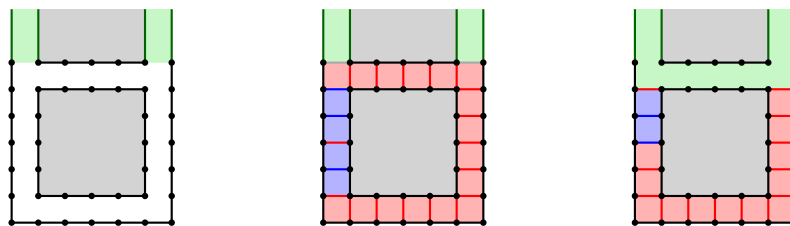
Proof. As we show in Lemma 7, optimal play of both Trudy and Fred results in them setting all the variables according to the rules described above in the first n moves. Afterwards the game enters a loony endgame with Fred in control.

If a variable gadget is set to TRUE, then there are three cycles left in the gadget: two overlapping cycles C_3 and C_4 , and the cycle C_1 connected to C_3 by a chain. Then Trudy can select C_1 and one of C_3 or C_4 to add to \mathcal{C}_{\max} .

If the variable gadget is set to FALSE, then there are still three cycles left in the gadget, but now these cycles are forming a chain where each consecutive pair of cycles is overlapping. Thus, if the output value is FALSE, then C_4 and the merged cycle of C_1 and C_2 can be added to \mathcal{C}_{\max} . On the other hand, if the output value is TRUE then C_4 cannot be in \mathcal{C}_{\max} , and from the remaining two cycles, only one can be selected to be added to \mathcal{C}_{\max} . ◀

3.5 Clause gadget

Finally, we describe a clause gadget that yields more boxes to Trudy if the signal entering the clause corresponds to TRUE. A clause gadget is simply an extra cycle extending the end of a wire gadget to an odd length. Figure 12 shows the gadget, and the two possible assignments of this gadget. Whenever the signal is TRUE, it is possible for Trudy to create a disjoint cycle in the gadget which gives her four boxes. If the signal is FALSE, Trudy can only make a chain in this gadget which yields only two boxes. Again, by considering the various cases that can occur under regular play, we obtain the following lemma.



■ **Figure 12** The clause gadget (left) yields four boxes to Trudy if the input signal is TRUE (middle), and only two boxes when the input is FALSE (right). Boxes highlighted in green belong to the last cycle in the adjacent wire.

► **Lemma 6.** *Under regular play, if the clause gadget is set to TRUE, then Trudy can add 1 cycle to \mathcal{C}_{\max} . Otherwise, under regular play, Trudy cannot add any cycle from the clause gadget to \mathcal{C}_{\max} .*

Proof. If the input signal to the clause gadget is TRUE, the adjacent cycle to the clause gadget is not in \mathcal{C}_{\max} . Therefore, a the cycle of the gadget can be added to \mathcal{C}_{\max} . When in the loony endgame, this cycle yields four boxes to Trudy after Fred makes a double-dealing move.

Otherwise, if the input signal is FALSE, the adjacent cycle is in \mathcal{C}_{\max} , and from the clause gadget only a chain is left. This chain yields only two boxes to Trudy after Fred makes a double-dealing move. ◀

4 Players' strategies and PSPACE-completeness

With the gadgets described above, we construct a Dots & Boxes instance for any G_{pos} (POS CNF) instance such that Trudy can win the Dots & Boxes instance if and only if she can win the corresponding G_{pos} (POS CNF) instance. We lay out the variable gadgets, attach a corresponding number of wire gadgets, pass the wires through or gadgets, using crossover gadgets to cross signals, and finally connect wires to the clause gadgets. An example of our construction is given in Figure 13 in the appendix.

The initial score we set to the Dots & Boxes instance depends on the number of gadgets of each type in the construction. By Lemma 1 the total score in the loony endgame depends on the number of disjoint cycles c , the number of boxes k with degree higher than two, and their total degree T . The configuration of the loony endgame, and thus the values k and T , is changed only when the variable gadgets are being assigned their values. We will argue below that, under regular play, exactly half of the variables are set to TRUE and half are set to FALSE. Thus the total values of k and T are the same, no matter which variables are assigned to which values. If Trudy can satisfy formula \mathcal{F} of the G_{pos} (POS CNF) game, she can claim some $c_{\max} = |\mathcal{C}_{\max}|$ cycles in the corresponding Dots & Boxes instance. Therefore, by Lemma 1, she can claim $4c_{\max} + T - 2k - 4$ boxes in the loony endgame, and $n/2$ boxes from the variables set to FALSE. Let N be the total number of unclaimed boxes in our Dots & Boxes instance. Then, Fred gets $N - n/2 - (4c_{\max} + T - 2k - 4)$ boxes. We set the initial scores of Trudy and Fred such that Trudy's final score is one larger than Fred's if she can satisfy \mathcal{F} . Otherwise, her score will be strictly less than Fred's.

Next, we describe the regular strategies for Trudy and Fred, both before the loony endgame is entered and in the loony endgame.

Regular strategies for Trudy and Fred in the loony endgame. We first discuss both strategies in the loony endgame, assuming that all variables have already been assigned a value as described in Section 3.4. As we argue below, Fred can always ensure that he is in control of the loony endgame. It is always beneficial for Fred to stay in control, as then all the chains and cycles in the loony endgame configuration yield at least as many boxes to him as to Trudy.

In the loony endgame, Trudy can choose which chains and cycles to open. To maximize her score, Trudy is going to select a maximum number of disjoint cycles C_{\max} in the loony endgame (see Lemma 1). This can be done by first making a loony move in all chains, to which Fred responds by claiming all but two boxes, finishing with a double-dealing move in order to stay in control. Afterwards, Trudy makes loony moves in the remaining cycles, to which Fred responds again by claiming all but four boxes, finishing with a double-dealing moves each time, except for in the final cycle.

Regular strategy for Trudy before the loony endgame. Trudy's strategy before the loony endgame is to set enough variable gadgets to TRUE in order to satisfy all the clauses. By Lemmas 1 and 6, Trudy gains more boxes from each satisfied clause. Therefore, the regular strategy for Trudy is to claim the boxes opened by Fred when setting variables to FALSE, and to set variables to TRUE, by using a loony move in a side chain of cycle C_2 of the variables. As we show in Lemma 7, if Fred deviates from setting variables to FALSE, and plays a loony move when there are non-loony moves available, Trudy can adopt Fred's regular strategy and dominate the rest of the game by ensuring that she ends up in control when the loony endgame is entered.

Regular strategy for Fred before the loony endgame. Fred's strategy is to ensure that he is in control when the loony endgame starts, and it can be described completely as responses to what Trudy does. By our assumption the number of variables in \mathcal{F} is even, thus initially the number of non-loony move pairs is even. Fred's strategy is then to keep the number of non-loony move pairs even at the start of every one of Trudy's turns. Then, once the number of non-loony moves reaches zero (and the loony endgame is reached), it is Trudy's turn, and Fred is in control. Specifically, Fred responds to Trudy's moves in the following way:

- If Trudy follows regular play and makes a loony move in a variable to set it to TRUE, then Fred simply claims all boxes in the chain opened by Trudy (without making a double-dealing move), and makes a non-loony move in another variable to set it to FALSE.
- If Trudy deviates from her strategy by making a non-loony move, setting a variable to FALSE, there must be at least one other non-loony move pair available to Fred. Therefore, Fred claims the box opened by Trudy, and makes a non-loony move, thereby setting another variable to FALSE. The number of non-loony pairs is again even at the start of Trudy's next turn.
- If Trudy deviates from her strategy by opening a chain with a loony move that does not remove a non-loony pair, Fred responds with claiming all but two (or four in case of a cycle) boxes and ends with a double-dealing move. The number of non-loony pairs remains even before Trudy's next turn.

Using this strategy, Fred can set a variable to FALSE each time Trudy sets a variable to any value, as well as gain control in the loony endgame.

Note that the order of moves in these strategies is not enforced. Trudy can play loony moves she would play in the loony endgame even if there are still non-loony moves available, as long as these moves do not interfere with the values set (or to be set) in the corresponding

variables. For Fred, we consider it part of his regular strategy to simply respond to these moves as if the game was already in the loony endgame, since otherwise he would be in danger of losing control. Indeed, if Fred does not make a double-dealing move, the number of non-loony moves will no longer be even at the start of Trudy's turn, and Fred loses control of the loony endgame. Thus, it is not more beneficial for any player to make a move in any other gadget than the variable gadgets while there are still variables that have not been set.

► **Lemma 7.** *Deviating from the regular strategies described above is sub-optimal for Fred and cannot be more beneficial for Trudy.*

Proof. Trivially, Trudy and Fred always claim open boxes before making their move, except when Fred makes double-dealing moves. Otherwise the opponent can claim these boxes next turn.

First, consider the regular strategies in the loony endgame. If Trudy deviates from her strategy and does not select the maximum number of disjoint cycles, by Lemma 1 her score will be too low and she loses the game. Therefore, the regular loony endgame strategy for Trudy as described above is optimal. If, at any point in the loony endgame, except for his last move, Fred does not make a double-dealing move, he loses control. Since being in control is always beneficial in our construction, this play is sub-optimal.

The regular strategies described for before the loony endgame are also optimal. Observe that, under the described strategies, the value-setting component of a variable yields the same number of boxes to Trudy independent of whether it is set to TRUE or to FALSE. Indeed, if it is set to TRUE, the component contains three boxes with degree 3, while setting the variable to TRUE does not give any boxes to Trudy; if the variable is set to FALSE, the component contains two boxes with degree 3, but setting the value gives Trudy one box. Thus, the value-setting component contributes the same number of points to Trudy's final score independent of the value.

If Trudy deviates from her strategy by making a non-loony move and setting a variable to FALSE, she loses one box to Fred. Furthermore, setting a variable to FALSE can never help Trudy to satisfy formula \mathcal{F} . Thus, such a move is sub-optimal.

If Trudy deviates from her strategy by making a loony move in any other gadget than the variable gadget, there are two options: either she makes a move that leads to the same score as the strategy described above, or she makes a move that contradicts the setting of the variables and reduces her total score. The former case does not have any bad repercussions for Trudy. Fred will respond with a double-dealing move, otherwise Trudy would take control of the endgame. Thus, we can reorder the sequence of Trudy's moves and assume that she first sets all the variables. However, in the latter case, the move reduces the number of possible disjoint cycles, and thus leads to Trudy's loss in the game. Therefore, deviating from the regular strategy is never more beneficial for Trudy.

If Fred deviates from his strategy before the loony endgame, then Trudy can adopt his strategy and ensure that the number of non-loony move pairs is even at the start of each of Fred's turn. Since, if Fred is not in control of the loony endgame, he loses the game, deviating from his strategy is not optimal. ◀

Combining the lemmas above, we obtain the main theorem.

► **Theorem 8.** *Dots & Boxes is PSPACE-complete.*

Proof. A game of Dots & Boxes is finished after a polynomial number of turns. Thus, all possible sequences of moves can be explored using polynomial sized memory. This implies that Dots & Boxes is in PSPACE.

We now show that Dots & Boxes is PSPACE-hard. Given a G_{pos} (POS CNF) formula \mathcal{F} , we construct a Dots & Boxes instance δ following the description above. We argue that Trudy can win \mathcal{F} if and only if Trudy can satisfy δ .

If Trudy can satisfy \mathcal{F} , then there must be a variable assignment following the G_{pos} (POS CNF) rules such that Trudy can ensure that every clause is connected to at least one variable which has been set to TRUE, regardless of what Fred does. Therefore, there can be at most $n/2$ variables that need to be set to TRUE by Trudy. Hence, Trudy can set the corresponding variable gadgets in δ to TRUE, and if needed set the remaining variables available to her to TRUE in any order. Thus, by Lemmas 2–6, Trudy can propagate the TRUE values down to all the clauses, that is, she can select the maximum number of disjoint cycles from all the gadgets, including all the clause gadgets, leading to the winning score in δ .

In order for Trudy to win δ , the set of disjoint cycles \mathcal{C}_{\max} that she selects must contain a cycle from every clause gadget, and the maximum number of cycles from all the other gadgets. By Lemmas 2–7, this can be done only if the output signals from each gadget conform to their input signals, and thus there must be a set of variable gadgets set to TRUE whose signal is propagated all the way down to all the clause gadgets. In δ Trudy and Fred have to alternate choosing which variable gadgets get set to TRUE and FALSE, respectively. Thus, if Trudy has a winning strategy in δ , no matter how Fred plays, she can always pick a subset of variable gadgets to assign, such that every clause gadget obtains a TRUE signal. This results in a winning strategy for Trudy to win the G_{pos} (POS CNF) game on \mathcal{F} .

Thus, Dots & Boxes is PSPACE-complete. ◀

5 Conclusion

We proved that Dots & Boxes is PSPACE-complete, resolving a long-standing open problem. There exist a number of other intriguing open problems related to Dots & Boxes. Does restricting the game to a $k \times n$ grid for a small k make the game easier? How large does k need to be to make the problem PSPACE-hard or even just NP-hard? These are challenging questions, given that even for a $1 \times n$ grid Dots & Boxes is not yet fully understood [12, 19, 25]. Another direction of further research is the complexity of variants of Dots & Boxes, in particular of *misère* Dots & Boxes [12], of Dots & Boxes under normal play (where the last player to move wins), of Dots & Boxes on other grids, or even of Dots & Boxes with more than two players as it was originally described by Lucas [29]. One variant that our result resolves is *Dots & Polygons*, since the reduction from Dots & Boxes to *Dots & Polygons* that was used to prove NP-hardness [8] now directly also shows PSPACE-hardness.

Our result can be interpreted as proving that *Strings and coins* restricted to grid graphs is PSPACE-complete. What is the complexity of *Strings and coins* on other restricted graph classes, for instance outerplanar graphs (which generalize $1 \times n$ grids)?

This may also be a good moment to revisit other games, which are known to be PSPACE-complete on general graphs, but for which the complexity on grid graphs is open. This, for instance, includes *NoGo*, *Fjords* (on hexagonal grids), *Cats-and-Dogs* and *GraphDistance*, which are known to be PSPACE-complete for planar graphs [9, 10].

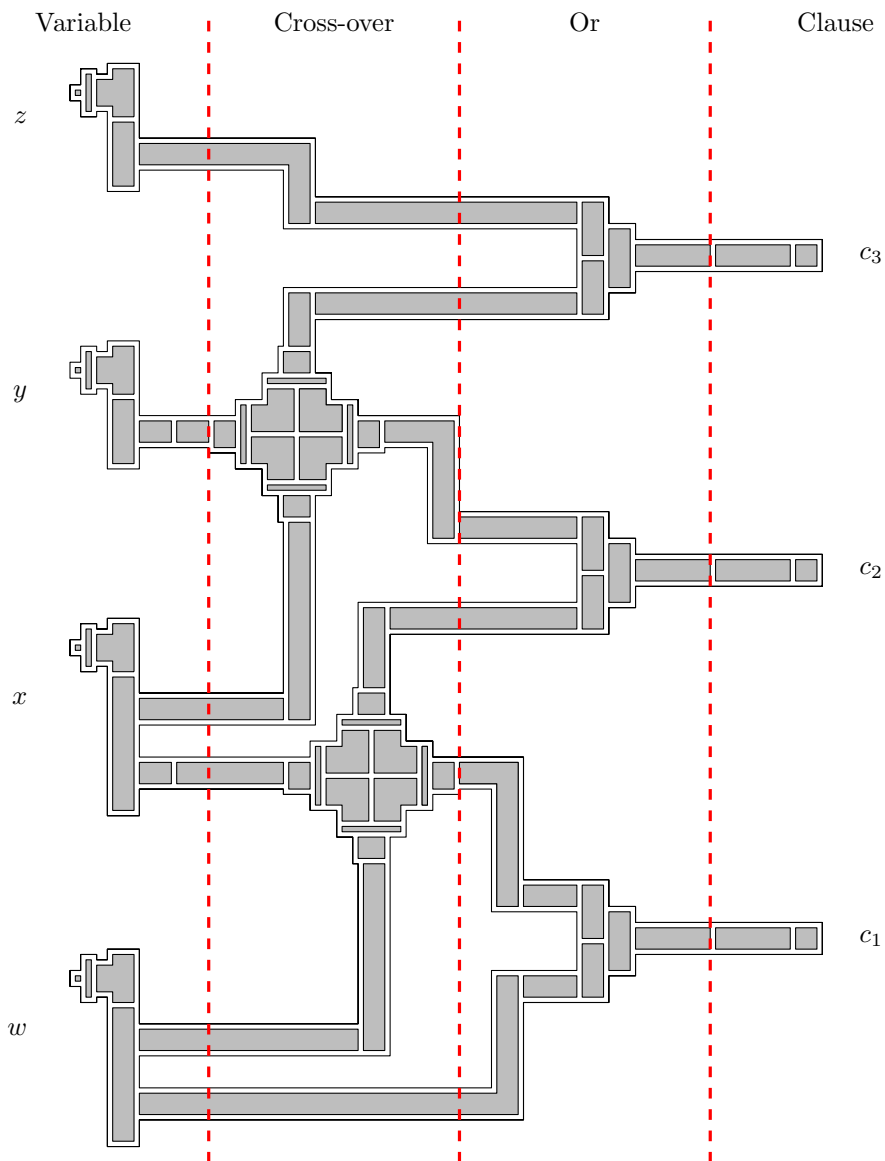
References

- 1 Oswin Aichholzer, David Bremner, Erik D. Demaine, Ferran Hurtado, Evangelos Kranakis, Hannes Krasser, Suneeta Ramaswami, Saurabh Sethia, and Jorge Urrutia. Games on triangulations. *Theoretical Computer Science*, 343(1):42–71, 2005. Game Theory Meets Theoretical Computer Science. doi:10.1016/j.tcs.2005.05.007.

- 2 Michael H. Albert, Richard J. Nowakowski, and David Wolfe. *Lessons in play: an introduction to combinatorial game theory*. CRC Press, 2019.
- 3 Joseph Barker and Richard Korf. Solving 4×5 dots-and-boxes. In *Proc. 25th AAAI Conference on Artificial Intelligence*, pages 1756–1757, 2011. doi:10.5555/2900423.2900680.
- 4 Joseph Barker and Richard Korf. Solving dots-and-boxes. In *Proc. 26th AAAI Conference on Artificial Intelligence*, pages 414–419, 2012. doi:10.5555/2900728.2900788.
- 5 Elwyn R. Berlekamp. *The Dots and Boxes Game: Sophisticated Child’s Play*. A K Peters/CRC Press, 2000.
- 6 Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. Chapter 16: Dots-and-boxes. In *Winning Ways for your Mathematical Plays*, volume 3, pages 541–584. A K Peters/CRC Press, 2nd edition, 2003.
- 7 Elwyn R. Berlekamp and Katherine Scott. Forcing your opponent to stay in control of a loony Dots-and-Boxes endgame. In R. J. Nowakowski, editor, *More Games of No Chance*, Proc. MSRI Workshop on Combinatorial Games, volume 42, pages 317–330. Cambridge University Press, 2002.
- 8 Kevin Buchin, Mart Hagedoorn, Irina Kostitsyna, Max van Mulken, Jolan Rensen, and Leo van Schooten. Dots & polygons (media exposition). In *Proc. 36th Internat. Sympos. Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 79:1–79:4, 2020. doi:10.4230/LIPIcs.SocG.2020.79.
- 9 Kyle Burke and Robert A. Hearn. PSPACE-complete two-color planar placement games. *International Journal of Game Theory*, 48(2):393–410, 2019. doi:10.1007/s00182-018-0628-8.
- 10 Kyle Burke, Silvia Heubach, Melissa A. Huggan, and Svenja Huntemann. Keeping your distance is hard. *CoRR*, 2016. arXiv:1605.06801.
- 11 Kevin Buzzard and Michael Ciere. Playing simple loony dots-and-boxes endgames optimally. *INTEGERS*, 14:2, 2014.
- 12 Sébastien Collette, Erik D. Demaine, Martin L. Demaine, and Stefan Langerman. Narrow misere Dots-and-Boxes. *Games of No Chance 4*, 63:57, 2015.
- 13 Erik D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In *International Symposium on Mathematical Foundations of Computer Science*, pages 18–33. Springer, 2001. doi:10.1007/3-540-44683-4_3.
- 14 Erik D. Demaine, Martin L. Demaine, Nicholas J.A. Harvey, Ryuhei Uehara, Takeaki Uno, and Yushi Uno. UNO is hard, even for a single player. *Theoretical Computer Science*, 521:51–61, 2014. doi:10.1016/j.tcs.2013.11.023.
- 15 Erik D. Demaine and Yevhenii Diomidov. Strings-and-Coins and Nimstring are PSPACE-complete. *CoRR*, 2021. arXiv:2101.06361.
- 16 Erik D. Demaine and Robert A. Hearn. Playing games with algorithms: algorithmic combinatorial game theory. In *Games of No Chance 3*, pages 3–56. Cambridge University Press, 2009. arXiv:cs/0106019.
- 17 David Eppstein. Computational complexity of games and puzzles. Last accessed on 06/05/2021. URL: <https://www.ics.uci.edu/~eppstein/cgt/hard.html>.
- 18 Gary William Flake and Eric B. Baum. Rush Hour is PSPACE-complete, or “Why you should generously tip parking lot attendants”. *Theoretical Computer Science*, 270(1):895–911, 2002. doi:10.1016/S0304-3975(01)00173-6.
- 19 Richard K. Guy and Richard J. Nowakowski. Unsolved problems in combinatorial games. In R. J. Nowakowski, editor, *More Games of No Chance*, Proc. MSRI Workshop on Combinatorial Games, volume 42, page 457–473. Cambridge University Press, 2002.
- 20 Robert A. Hearn. *Games, puzzles, and computation*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.
- 21 Robert A. Hearn and Erik D. Demaine. *Games, puzzles, and computation*. CRC Press, 2009.
- 22 Takashi Horiyama, Takashi Iizuka, Masashi Kiyomi, Yoshio Okamoto, Ryuhei Uehara, Takeaki Uno, Yushi Uno, and Yukiko Yamauchi. Sankaku-tori: An old western-japanese game played

- on a point set. *Journal of Information Processing*, 25:708–715, 2017. doi:10.2197/ipsjjip.25.708.
- 23 Ming Yu Hsieh and Shi-Chun Tsai. On the fairness and complexity of generalized k -in-a-row games. *Theoretical Computer Science*, 385(1-3):88–100, 2007. doi:10.1016/j.tcs.2007.05.031.
 - 24 Shigeki Iwata and Takumi Kasai. The Othello game on an $n \times n$ board is PSPACE-complete. *Theoretical Computer Science*, 123(2):329–340, 1994. doi:10.1016/0304-3975(94)90131-7.
 - 25 Adam S. Jobson, Levi Sledd, Susan Calcote White, and D. Jacob Wildstrom. Variations on narrow dots-and-boxes and dots-and-triangles. *INTEGERS*, 17:#G2:1–8, 2017.
 - 26 Will Johnson. The combinatorial game theory of well-tempered scoring games. *International Journal of Game Theory*, 43(2):415–438, 2014. doi:10.1007/s00182-013-0386-6.
 - 27 Anthony Knittel, Terry Bossomaier, and Allan Snyder. Concept accessibility as basis for evolutionary reinforcement learning of dots and boxes. In *IEEE Symposium on Computational Intelligence and Games*, pages 140–145. IEEE, 2007. doi:10.1109/CIG.2007.368090.
 - 28 David Lichtenstein and Michael Sipser. GO is polynomial-space hard. *Journal of the ACM*, 27(2):393–401, 1980. doi:10.1145/322186.322201.
 - 29 Édouard Lucas. *Récréations mathématiques*, volume 2. Gauthier-Villars et fils, 1883.
 - 30 Henry Meyniel and Jean-Pierre Roudneff. The vertex picking game and a variation of the game of dots and boxes. *Discrete Mathematics*, 70:311–313, 1988. doi:10.1016/0012-365X(88)90007-6.
 - 31 Richard J. Nowakowski. . . ., Welter’s game, Sylver coinage, dots-and-boxes, In R. K. Guy, editor, *Combinatorial Games*, Proc. Symposia in Applied Mathematics, volume 43, pages 155–182. AMS, 1991. doi:10.1090/psapm/043/1095544.
 - 32 Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978. doi:10.1016/0022-0000(78)90045-4.
 - 33 Aaron N. Siegel. *Combinatorial game theory*, volume 146 of *Graduate Studies in Mathematics*. American Mathematical Society, 2013. doi:10.1090/gsm/146.
 - 34 Julian West. Championship-level play of dots-and-boxes. In R. J. Nowakowski, editor, *Games of No Chance*, Proc. MSRI Workshop on Combinatorial Games, volume 29, pages 79–84. Cambridge University Press, 1996.
 - 35 Yimeng Zhuang, Shuqin Li, Tom Vincent Peters, and Chenguang Zhang. Improving monte-carlo tree search for dots-and-boxes with a novel board representation and artificial neural networks. In *IEEE Conference on Computational Intelligence and Games*, pages 314–321. IEEE, 2015. doi:10.1109/CIG.2015.7317912.

A Example game



■ **Figure 13** Example reduction from the G_{pos} (POS CNF) formula $(w \vee x) \wedge (w \vee y) \wedge (x \vee z)$. The construction can be divided into four sections: a variable, crossover, or, and clause section. Each section contains only the corresponding gadgets and wire gadgets that connect different gadgets together.

B Omitted proofs

► **Lemma 1.** *Let the configuration of a loony endgame contain k boxes with degree higher than 2, let T be the sum of the degrees of these boxes, and let c be the maximum number of disjoint cycles in the configuration. Then, the player who is not in control can claim at most $4c + T - 2k - 4$ boxes.*

Proof. Let Fred be in control of the game. To simplify the argument, w.l.o.g., we assume that the last move made by Trudy is made in a cycle. Let c denote the number loony moves made by Trudy in a disjoint cycle and let ℓ be the number of loony moves made by Trudy in chains. All but the last loony move in a disjoint cycle or chain yield 4 or 2 boxes for Trudy, respectively. Thus, the score gained by Trudy in the loony endgame is

$$4c + 2\ell - 4.$$

Consider the dual graph $G = (V, E)$ to the Dots & Boxes instance. In it, a node corresponds to a box, and an edge connects two nodes if the two corresponding adjacent boxes do not have a line drawn between them. Suppose G has k nodes with degree higher than 2. We define T to be the sum of the degrees of these nodes:

$$T = \sum_{\{v \in V \mid \text{degree}(v) > 2\}} \text{degree}(v).$$

A loony move on a disjoint cycle does not change T , since all disjoint cycles only contain boxes of degree 2. A loony move on a chain, however, decreases the degree of the box at both ends of the chain by 1. Furthermore, whenever the degree of a box reduces from 3 to 2 the degree of this box is no longer counted in T . Thus

$$T = 2\ell + 2k,$$

which means the score for Trudy will be

$$4c + T - 2k - 4.$$

Since T and k are fixed, the score is maximized when the number of loony moves in disjoint cycles is maximized. ◀

Uncertain Curve Simplification

Kevin Buchin   

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Maarten Löffler  

Department of Information and Computing Sciences, Utrecht University, The Netherlands

Aleksandr Popov   

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Marcel Roeloffzen  

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

We study the problem of polygonal curve simplification under uncertainty, where instead of a sequence of exact points, each uncertain point is represented by a region which contains the (unknown) true location of the vertex. The regions we consider are disks, line segments, convex polygons, and discrete sets of points. We are interested in finding the shortest subsequence of uncertain points such that no matter what the true location of each uncertain point is, the resulting polygonal curve is a valid simplification of the original polygonal curve under the Hausdorff or the Fréchet distance. For both these distance measures, we present polynomial-time algorithms for this problem.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Curves, Uncertainty, Simplification, Fréchet Distance, Hausdorff Distance

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.26

Related Version *Full Version:* <https://arxiv.org/abs/2103.09223> [12]

Funding *Maarten Löffler:* Partially supported by the Dutch Research Council (NWO) under project no. 614.001.504.

Aleksandr Popov: Supported by the Dutch Research Council (NWO) under project no. 612.001.801.

Marcel Roeloffzen: Supported by the Dutch Research Council (NWO) under project no. 628.011.005.

1 Introduction

In this paper, we study the topic of curve simplification under uncertainty. There are many classical algorithms dealing with curve simplification with different distance metrics; however, it is typically assumed that the locations of points making up the curves are known precisely, which often does not suit real-life data. An example highlighting the necessity of taking uncertainty into account comes with GPS data, where each measured location is inherently imprecise, and the real location is likely to be within a certain distance from the measurement. This imprecision can be modelled as a disk (or some other shape if the GPS signals are blocked or reflected by rocks, buildings, etc.). Curve simplification is used to reduce the noise-to-signal ratio in the trajectory data before applying other algorithms or when storing large amounts of data. In both cases modelling uncertainty could reduce the error introduced by simplifying imprecise curves while maintaining a short, efficient representation of the data.

There is a large volume of foundational work on curve simplification [4], including work on vertex-constrained simplification, such as the algorithms by Ramer and by Douglas and Peucker [17, 34] using the Hausdorff distance, by Agarwal et al. [3] using the Fréchet distance, by Imai and Iri [23] using either, and various improvements and related approaches [7, 8, 10, 16, 21, 22, 31, 36]. The Imai–Iri algorithm involves computing the *shortcut graph*, which captures all the possible simplifications of a curve, and then finding a path through the graph



© Kevin Buchin, Maarten Löffler, Aleksandr Popov, and Marcel Roeloffzen;
licensed under Creative Commons License CC-BY 4.0

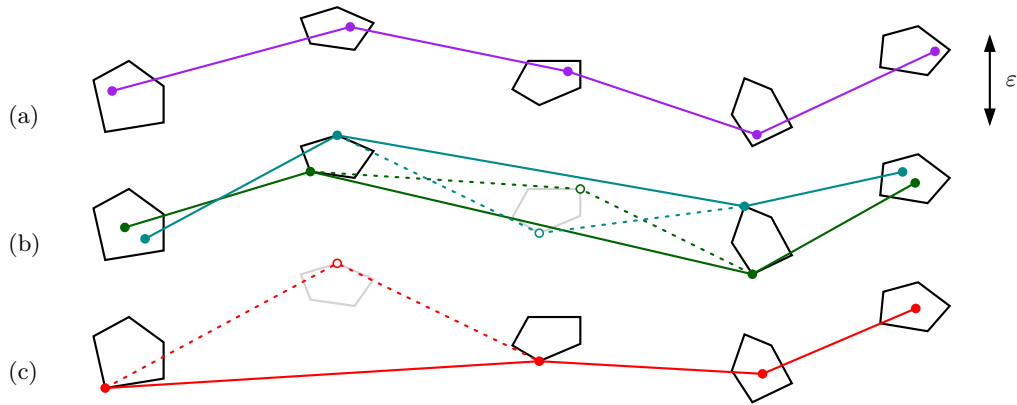
46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 26; pp. 26:1–26:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** (a) An uncertain curve modelled with convex polygons and a realisation. (b) A valid simplification under the Hausdorff distance with the threshold ϵ : for every realisation, the subsequence is within distance ϵ from the full sequence. (c) An invalid simplification: there is a realisation for which the subsequence is not within distance ϵ from the full sequence.

with minimal edge count from the start to the end node, yielding the shortest simplification. We adapt this approach to the setting with uncertainty. It seems natural to apply disk stabbing to test shortcuts [22]; we discuss why this does not work in our setting in Section 3.

There are recent advances in the study of uncertainty in computational geometry, with work on optimising various measures on uncertain points [24, 25, 26, 28, 30], triangulations [11, 29, 37], visibility in uncertain polygons [15], and other problems [1, 2, 18, 19, 20, 27, 32, 35]. There is work by Ahn et al. [5], and, more recently, by Buchin et al. [9, 33] on various minimisation and maximisation variants of curve similarity with the Fréchet distance under uncertainty, and other work combining trajectory analysis and uncertainty [6, 13, 14]. To our knowledge, there is no previous work studying curve simplification under uncertainty.

We use the locational model for uncertainty: we know that each point exists, but not its exact location. It can be modelled as a discrete set of points, of which one is the true location; this model uses *indecisive* points. We also use *imprecise* points, modelled as compact continuous sets, such as disks, line segments, or convex polygons; the true location is one unknown point from the set. An *uncertain curve* is a sequence of uncertain points of the same kind. A *realisation* of an uncertain curve is a polygonal curve obtained by taking one point from each uncertain point. We solve the following problem (see Figure 1): *given an uncertain curve as a sequence of n uncertain points, find the shortest subsequence of the points such that for any realisation of the curve, the realisation restricted to the subsequence is a valid simplification*. We give efficient algorithms for this problem for the Hausdorff and the Fréchet distance. They run in $\mathcal{O}(n^3)$ time for uncertainty modelled with disks or line segments and in $\mathcal{O}(n^3k^3)$ time for convex polygons and indecisive points with k vertices.

2 Preliminaries

Denote¹ $[n] \stackrel{\text{def}}{=} \{1, 2, \dots, n\}$ for any $n \in \mathbb{N}^{>0}$. Given two points $p, q \in \mathbb{R}^2$, denote their Euclidean distance with $\|p - q\|$.

¹ We use $:=$ and $=$ to denote assignment, $\stackrel{\text{def}}{=}$ for equivalent quantities in definitions or to point out equality by earlier definition, and \equiv in other contexts. We also use \equiv , but its usage is always explained.

Denote a *sequence* of points in \mathbb{R}^2 with $\pi = \langle p_1, \dots, p_n \rangle$. For only two points $p, q \in \mathbb{R}^2$, we also write pq instead of $\langle p, q \rangle$. Denote a subsequence of a sequence π from index i to j with $\pi[i : j] = \langle p_i, p_{i+1}, \dots, p_j \rangle$. This notation can also be applied if we interpret π as a *polygonal curve* on n vertices (of *length* n). It is defined by linearly interpolating between the successive points in the sequence and can be seen as a continuous function, for $i \in [n - 1]$ and $\alpha \in [0, 1]$: $\pi(i + \alpha) = (1 - \alpha)p_i + \alpha p_{i+1}$.

We also introduce the notation for the order of points along a curve. Let $p := \pi(a)$ and $q := \pi(b)$ for $a, b \in [1, n]$. Then $p \prec q$ iff $a < b$, $p \preceq q$ iff $a \leq b$, and $p \equiv q$ iff $a = b$. Note that we can have $p = q$ for $a \neq b$ if the curve intersects itself.

Finally, given points $p, q, r \in \mathbb{R}^2$, define the distance from p to the segment qr as $d(p, qr) \stackrel{\text{def}}{=} \min_{t \in qr} \|p - t\|$.

An *uncertainty region* $U \subset \mathbb{R}^2$ describes a possible location of a true point: it has to be inside the region, but there is no information as to where exactly. We use several uncertainty models, so the regions U are of different shape. An *indecisive point* is a form of an uncertain point where the uncertainty region is represented as a discrete set of points, and the true point is one of them: $U = \{p^1, \dots, p^k\}$, with $k \in \mathbb{N}^{>0}$ and $p^i \in \mathbb{R}^2$ for all $i \in [k]$. *Imprecise points* are modelled with uncertainty regions that are compact continuous sets. In particular, we consider *disks* and *polygonal closed convex sets*. We denote a disk with the centre $c \in \mathbb{R}^2$ and the radius $r \in \mathbb{R}^{\geq 0}$ as $D(c, r)$. Formally, $D(c, r) \stackrel{\text{def}}{=} \{p \in \mathbb{R}^2 \mid \|p - c\| \leq r\}$. Define a *polygonal closed convex set (PCCS)* as a closed convex set with bounded area that can be described as the intersection of a *finite* number of closed half-spaces. Note that this definition includes both convex polygons and line segments (in 2D). Given a PCCS U , let $V(U)$ denote the set of vertices of U , i.e. vertices of a convex polygon or endpoints of a line segment.

We call a sequence of uncertainty regions an *uncertain curve*: $\mathcal{U} = \langle U_1, \dots, U_n \rangle$. If we pick a point from each uncertainty region of \mathcal{U} , we get a polygonal curve π that we call a *realisation* of \mathcal{U} and denote it with $\pi \in \mathcal{U}$. That is, if for some $n \in \mathbb{N}^{>0}$ we have $\pi = \langle p_1, \dots, p_n \rangle$ and $\mathcal{U} = \langle U_1, \dots, U_n \rangle$, then $\pi \in \mathcal{U}$ if and only if $p_i \in U_i$ for all $i \in [n]$.

Suppose we are given a polygonal curve $\pi = \langle p_1, \dots, p_n \rangle$, a threshold $\varepsilon \in \mathbb{R}^{>0}$, and a curve built on the subsequence of vertices of π for some set $I = \{i_1, \dots, i_\ell\} \subseteq [n]$, i.e. $\sigma = \langle p_{i_1}, \dots, p_{i_\ell} \rangle$ with $i_j < i_{j+1}$ for all $j \in [\ell - 1]$ and $\ell \leq n$. We call σ an ε -*simplification* of π if for each segment $\langle p_{i_j}, p_{i_{j+1}} \rangle$, we have $\delta(\langle p_{i_j}, p_{i_{j+1}} \rangle, \pi[i_j : i_{j+1}]) \leq \varepsilon$, where δ denotes some distance measure, e.g. the Hausdorff or the Fréchet distance.

The *Hausdorff distance* between two sets $P, Q \subset \mathbb{R}^2$ is defined as

$$d_H(P, Q) \stackrel{\text{def}}{=} \max \left\{ \sup_{p \in P} \inf_{q \in Q} \|p - q\|, \sup_{q \in Q} \inf_{p \in P} \|p - q\| \right\}.$$

For two polygonal curves π and σ in \mathbb{R}^2 , since π and σ are closed and bounded, we get

$$d_H(\pi, \sigma) = \max \left\{ \max_{p \in \pi} \min_{q \in \sigma} \|p - q\|, \max_{q \in \sigma} \min_{p \in \pi} \|p - q\| \right\}.$$

The *Fréchet distance* is often described through an analogy with a person and a dog walking along their respective curves without backtracking, where the Fréchet distance is the shortest leash needed for such a walk. Formally, consider a set of *reparametrisations* Φ_ℓ of length ℓ , defined as continuous non-decreasing surjective functions $\phi : [0, 1] \rightarrow [1, \ell]$. Given two polygonal curves π and σ of lengths m and n , respectively, we can define the Fréchet distance as

$$d_F(\pi, \sigma) \stackrel{\text{def}}{=} \inf_{\alpha \in \Phi_m, \beta \in \Phi_n} \max_{t \in [0, 1]} \|\pi(\alpha(t)) - \sigma(\beta(t))\|.$$

We refer to the pair of reparametrisations as an *alignment*. We often consider the Fréchet distance between a curve $\pi = \langle p_1, \dots, p_n \rangle$ and a line segment $p_1 p_n$, for some $n \in \mathbb{N}^{\geq 3}$. In this setting, the alignment can be described in a more intuitive way; see also Figure 2. It can be described as a sequence of locations on the line segment with which the vertices of the curve are aligned, $\langle s_2, \dots, s_{n-1} \rangle$, where $s_i \in [1, 2]$ for all $i \in \{2, \dots, n-1\}$ and $s_i \leq s_{i+1}$ for all $i \in \{2, \dots, n-2\}$. To see that, assign $s_1 := 1$ and $s_n := 2$ and construct a helper reparametrisation $\phi : [0, 1] \rightarrow [1, n]$, defined as $\phi(t) = (n-1) \cdot t + 1$ for any $t \in [0, 1]$. Construct another reparametrisation $\psi : [1, n] \rightarrow [1, 2]$, defined as

$$\psi(t) = \begin{cases} s_{\lfloor t \rfloor} \cdot (1 - t + \lfloor t \rfloor) + s_{\lfloor t \rfloor + 1} \cdot (t - \lfloor t \rfloor) & \text{if } t \in [1, n), \\ s_n & \text{if } t = n. \end{cases}$$

Note that ϕ and $\psi \circ \phi$ satisfy the definition of reparametrisations for π and $p_1 p_n$, respectively.

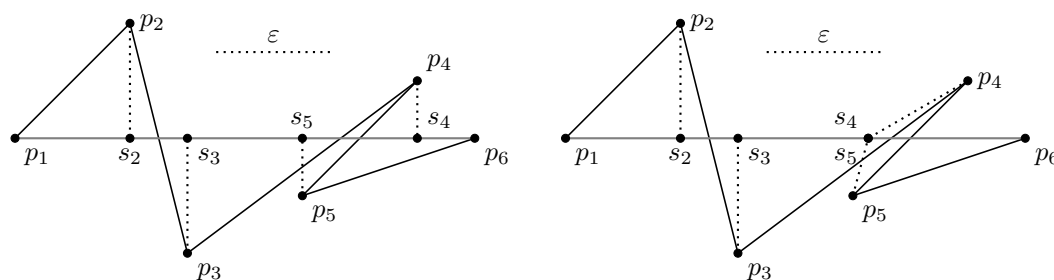
We also define an *alignment* between a curve and a line segment for the Hausdorff distance (see Figure 2). It represents the map from the curve to the line segment, where each point on the curve is mapped to the closest point on the line segment. It is given by a sequence $\langle s_1, \dots, s_n \rangle$, where $s_i \in [1, 2]$ for all $i \in [n]$, such that $p_1 p_n(s_i) = \operatorname{argmin}_{p' \in p_1 p_n} \|p' - p_i\|$. In other words, $p_1 p_n(s_i)$ is the closest point to p_i for all $i \in [n]$; as we discuss in Appendix A.1, the Hausdorff distance is realised as the distance between p_i and $p_1 p_n(s_i)$ for some $i \in [n]$. Therefore, establishing such an alignment and checking that $\|p_1 p_n(s_i) - p_i\| \leq \varepsilon$ for all $i \in [n]$ allows us to check that $d_H(\pi, p_1 p_n) \leq \varepsilon$ for some $\varepsilon \in \mathbb{R}^{>0}$.

We are discussing the following problem: given an uncertain curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $n \in \mathbb{N}^{\geq 3}$ and $U_i \subset \mathbb{R}^2$ for all $i \in [n]$, and the threshold $\varepsilon \in \mathbb{R}^{>0}$, find a minimal-length subsequence $\mathcal{U}' = \langle U_{i_1}, \dots, U_{i_\ell} \rangle$ of \mathcal{U} with $\ell \leq n$, such that for any realisation $\pi \in \mathcal{U}$, the corresponding realisation $\pi' \in \mathcal{U}'$ forms an ε -simplification of π under some distance measure δ . We solve this problem for the Hausdorff and the Fréchet distance for uncertainty modelled with indecisive points, line segments, disks, and convex polygons.

3 Overview of the Approach

We first present the summary of our approach. On the highest level, we use the *shortcut graph*. Each uncertain point of a curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ corresponds to a vertex. An edge connects two vertices i and j if and only if the distance between any realisation of $\mathcal{U}[i : j]$ and the corresponding line segment from U_i to U_j is below the threshold. The path with the fewest edges from vertex 1 to vertex n then corresponds to the simplification using fewest uncertain points. So, we construct the shortcut graph and find the shortest path between two vertices. The key idea is that we find shortcuts that are valid for *all* realisations, so any sequence of shortcuts can be chosen.

In order to construct the shortcut graph, we need to check whether an edge should be added to the graph, i.e. whether a shortcut is *valid*. It is natural to think that shortcut testing can be solved by disk stabbing with disks of suitable radius, as in the work by Guibas et al. [22]. The idea would then be, given the distance threshold ε , to replace the uncertainty regions with the intersection of ε -disks over all the points of a region; this way we would e.g. replace disks of radius r by disks of radius $\varepsilon - r$, and then check if a shortcut stabs these regions. However, except for disks, this approach does not work – the reader can see this by trying to apply the method on an uncertainty region shaped as a long line segment (or a skinny convex polygon) that is parallel to the potential shortcut line segment. The intersection of ε -disks may be empty, while clearly one can create an alignment for both the Hausdorff and the Fréchet distance. For disks the approach is more suitable; however, when



■ **Figure 2** Left: Alignment for the Hausdorff distance. Right: Alignment for the Fréchet distance. In both cases, the alignment is described as the sequence $\langle s_1 := p_1, s_2, s_3, s_4, s_5, s_6 := p_6 \rangle$.

testing a shortcut, the first and the last disk of a shortcut fulfil a different function than the intermediate disks. This means that we can rephrase the problem for the intermediate disks of a shortcut as disk stabbing, but not for the first and the last disk, as the quantifiers in the problem are different. Furthermore, the work by Guibas et al. [22] does not provide running time guarantees for disks of different radii, and the initialisation in their approach is not applicable in our setting with no restriction on disk intersections. So, we need to use a different approach to test shortcuts.

The approach is different for the Hausdorff and the Fréchet distance and for each uncertainty model. For the first and the last uncertain point of the shortcut, we state in Section 5 that there are several critical pairs of realisations that need to be tested explicitly, and then for any other pair of realisations, we know that the distance is also below the threshold. Testing each pair corresponds to finding the distance between a precise line segment and any realisation of an uncertain curve; we discuss this in Section 4 and show the procedures to do this in detail in Appendix A.

► **Theorem 1.** *We can find the shortest vertex-constrained simplification of an uncertain curve, such that for any realisation the simplification is valid, both for the Hausdorff and the Fréchet distance, in time $\mathcal{O}(n^3)$ for uncertainty modelled with disks and line segments, and in time $\mathcal{O}(n^3k^3)$ for uncertainty modelled with indecisive points and convex polygons, where k is the number of options or vertices and n is the length of the curve.*

4 Shortcut Testing: Intermediate

Here we discuss testing a shortcut with the first and the last points fixed, i.e. we want to check $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \delta(\pi, p_1 p_n) \leq \varepsilon$ for $\delta := d_H$ and $\delta := d_F$. We can do so in linear time in all the models; here we show the intuitive explanation, and we treat this topic in detail in Appendix A. We solve the following problem.

► **Problem 2.** Given an uncertain curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ on $n \in \mathbb{N}^{\geq 3}$ uncertain points in \mathbb{R}^2 , as well as realisations $p_1 \in U_1, p_n \in U_n$, check if the largest Hausdorff or Fréchet distance between \mathcal{U} and its one-segment simplification is below a threshold $\varepsilon \in \mathbb{R}^{>0}$ for any realisation with the fixed start and end points, i.e. for $\delta := d_H$ or $\delta := d_F$, verify

$$\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \delta(\pi, p_1 p_n) \leq \varepsilon.$$

Hausdorff distance. It is a well-known fact that the Hausdorff distance between the curve and the line segment that simplifies that curve is the largest distance from a vertex of the curve to the line segment, so $d_H(\pi, \langle \pi(1), \pi(n) \rangle) = \max_{i \in [n]} d(\pi(i), \langle \pi(1), \pi(n) \rangle)$ for a polygonal curve π of length n . (See Figure 2.) We can use the same idea in the uncertain setting; however, for indecisive curves, we can choose any realisation for each intermediate point, so we need to test all of them, so we need the largest distance from any realisation of any indecisive point to the line segment. Then for indecisive points, given a curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$, we have $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \delta(\pi, p_1 p_n) = \max_{i \in [n]} \max_{j \in [k]} d(p_i^j, p_1 p_n)$. For disks, line segments, and convex polygons, the key point is the same: all of the realisations of the intermediate points need to be close enough to the given line segment. For disks, we can simply check the furthest points, which are one radius further away from the line segment than the disk centre. For line segments and convex polygons, it suffices to test all the vertices.

Fréchet distance. For the Fréchet distance, there is also an intuitive procedure in the precise setting [22, Lemma 8]. We can align each vertex from the curve with the earliest possible point in the line segment. Each next point cannot be aligned before the previous points, so choosing the earliest possible alignment point maximises the possibilities for the remainder of the curve. (See Figure 2: s_4 is as close as possible to p_1 .) We use the same approach in the uncertain setting; however, for indecisive points, as any realisation of a point is possible, we need to choose the realisation that pushes the earliest alignment forward the most, as this is the most restrictive realisation for the remainder of the curve. In more detail, we iteratively find the value for s_i . Given s_{i-1} , we find the earliest t_i^j along the segment for each realisation p_i^j of U_i , such that $\|t_i^j - p_i^j\| \leq \varepsilon$ and $s_{i-1} \preceq t_i^j$. Then we pick $s_i := \max_{j \in [k]} t_i^j$, in terms of \preceq . We continue this procedure until the end of the segment, starting with $s_1 := p_1$ and assigning $s_n := p_n$. In one direction, the sequence of s_i we find corresponds to a possible realisation; in the other direction, we can see that for any $i \in \{2, \dots, n-1\}$, we have $s_{i-1} \preceq t_i^j \preceq s_i$ for all $j \in [k]$; so for any other realisation the alignment is in order, as well. We can show for line segments and convex polygons that we again only need to focus on the vertices. For disks, we instead reframe the problem as that of disk stabbing. Instead of testing closeness from all points of some $D(c, r)$ to the line segment, we can check if the line segment stabs $D(c, \varepsilon - r)$ for the threshold ε . Then the correct alignment order corresponds to picking points inside disks in order. Again, choosing the earliest possible one is key.

5 Shortcut Testing: All Points

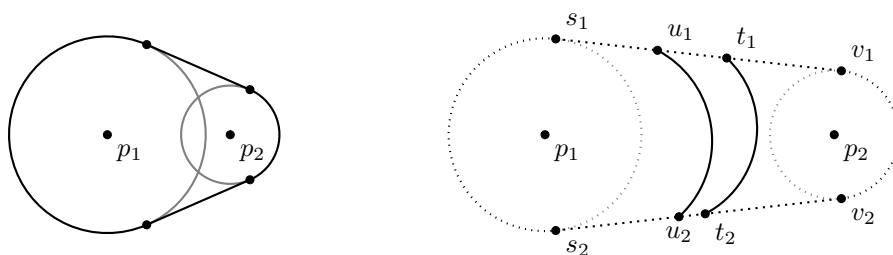
In the previous section, we have covered testing a shortcut, given that the first and the last points are fixed. Here we remove the restriction on the endpoints.

► **Problem 3.** Given an uncertain curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ on $n \in \mathbb{N}^{\geq 3}$ uncertain points in \mathbb{R}^2 , check if the largest Hausdorff or Fréchet distance between \mathcal{U} and its one-segment simplification is below a threshold $\varepsilon \in \mathbb{R}^{>0}$ for any realisation, i.e. for $\delta := d_H$ or $\delta := d_F$, verify $\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.

For the indecisive points, we can simply check all pairs from $U_1 \times U_n$; this is quite easy to show.

► **Lemma 4.** Given $n, k \in \mathbb{N}^{>0}$, $n \geq 3$, and $\delta := d_H$ or $\delta := d_F$, for any indecisive curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$ and $p_i^j \in \mathbb{R}^2$ for all $i \in [n]$, $j \in [k]$, we have

$$\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) = \max_{a \in [k]} \max_{b \in [k]} \max_{\sigma \in \mathcal{U}, \sigma(1) \equiv p_1^a, \sigma(n) \equiv p_n^b} \delta(\sigma, p_1^a p_n^b).$$



■ **Figure 3** Left: Illustration for Observation 5. The convex hull of the disks is highlighted in black. The order in which the outer tangents touch the disks is the same. Right: Illustration for Definition 6. Here O_1 (t_1 to t_2) is to the right of O_2 (u_1 to u_2).

Proof. We can derive

$$\begin{aligned}
 & \max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \\
 & \quad \{\text{Def. } \Subset\} \\
 & = \max_{p_1 \in U_1, \dots, p_n \in U_n} \delta(\langle p_1, \dots, p_n \rangle, p_1 p_n) \\
 & = \max_{p_1 \in U_1} \max_{p_n \in U_n} \max_{p_2 \in U_2, \dots, p_{n-1} \in U_{n-1}} \delta(\langle p_1, \dots, p_n \rangle, p_1 p_n) \\
 & \quad \{\text{Def. } \Subset\} \\
 & = \max_{p_1 \in U_1} \max_{p_n \in U_n} \max_{\sigma \in \mathcal{U}, \sigma(1) \equiv p_1, \sigma(n) \equiv p_n} \delta(\sigma, p_1 p_n) \\
 & = \max_{a \in [k]} \max_{b \in [k]} \max_{\sigma \in \mathcal{U}, \sigma(1) \equiv p_1^a, \sigma(n) \equiv p_n^b} \delta(\sigma, p_1^a p_n^b),
 \end{aligned}$$

as was to be shown. ◀

That is to say, for either Hausdorff or Fréchet distance we can simply test the shortcut using the corresponding procedure from Lemma 16 or Lemma 20, and do so for each combination of the start and end points. We can then test an indecisive shortcut of length n overall in time $\mathcal{O}(k^2 \cdot nk) = \mathcal{O}(nk^3)$.

We now proceed to show the approach for disks and polygonal closed convex sets. The procedure is the same for the Hausdorff and the Fréchet distance, but differs between disks and PCCSs, since disks have some convenient special properties.

5.1 Disks

► **Observation 5.** *Suppose we are given two non-degenerate disks $D_1 := D(p_1, r_1)$ and $D_2 := D(p_2, r_2)$ with $D_1 \not\subseteq D_2$ and $D_2 \not\subseteq D_1$. We make the following observations.*

- *There are exactly two outer tangents to the disks, and the convex hull of $D_1 \cup D_2$ consists of an arc from D_1 , an arc from D_2 , and the outer tangents.*
- *Assume the lines of the outer tangents intersect. When viewed from the intersection point, the order in which the tangents touch the disks is the same, i.e. either both first touch D_1 and then D_2 , or the other way around. If the lines are parallel, the same statement holds when viewed from points on the tangent lines at infinity. (See Figure 3.)*

To see that the second point is true, note that the distance from the intersection point to the tangent points of a disk is the same for both tangent lines. These observations mean that we can restrict our attention to the area bounded by the outer tangents and define an ordering in the resulting strip.

► **Definition 6.** Given two distinct non-degenerate disks $D_1 := D(p_1, r_1)$ and $D_2 := D(p_2, r_2)$, consider a strip defined by the lines that form the outer tangents to the disks. Assume we have two circular arcs O_1, O_2 that intersect both tangents and lie inside the strip. Define s_1 and v_1 to be the points where one of the tangents touches D_1 and D_2 , respectively, and let t_1 and u_1 be the points where O_1 and O_2 intersect that tangent, respectively. Define the order on the tangents from D_1 to D_2 , so $s_1 \prec v_1$. Define points s_2, t_2, u_2, v_2 similarly for the other tangent. We say that O_2 is to the right of O_1 if either $t_i = u_i$ for $i \in \{1, 2\}$ and the radius of O_1 is larger than that of O_2 ; or if otherwise $t_i \preceq u_i$ for $i \in \{1, 2\}$ and O_1 and O_2 do not properly intersect. We say that O_2 is to the left of O_1 if either $t_i = u_i$ for $i \in \{1, 2\}$ and the radius of O_1 is smaller than that of O_2 ; or if otherwise $u_i \preceq t_i$ for $i \in \{1, 2\}$ and O_1 and O_2 do not properly intersect. (See Figure 3 for a visual interpretation.)

We state the main result: it suffices to check the tangents to the first and the last disk and the order of the intermediate disks.

► **Lemma 7.** Given $n \in \mathbb{N}^{\geq 3}$, for any imprecise curve modelled with disks $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = D(c_i, r_i)$ for all $i \in [n]$ and $c_i \in \mathbb{R}^2$, $r_i \in \mathbb{R}^{\geq 0}$ for all $i \in [n]$, and assuming $U_1 \neq U_n$, we have with $\delta \in \{d_H, d_F\}$ that $\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$ if and only if both of the following are true:

$$\blacksquare \max \left\{ \max_{\pi \in \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} \delta(\pi, st), \max_{\pi \in \mathcal{U}, \pi(1) \equiv u, \pi(n) \equiv v} \delta(\pi, uv) \right\} \leq \varepsilon,$$

where $s, u \in U_1$, $t, v \in U_n$, and st and uv are the outer tangents to $U_1 \cup U_n$;

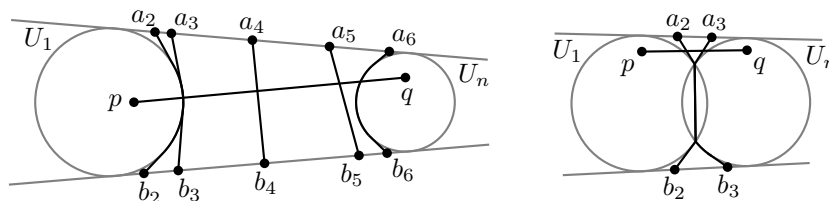
► for each $i \in \{2, \dots, n-1\}$, the right arc of the disk $D(c_i, \varepsilon - r_i)$ bounded by the intersection points with the tangent lines is to the right of the right arc of U_1 and the left arc of the disk $D(c_i, \varepsilon - r_i)$ is to the left of the left arc of U_n .

Proof. We first prove the claim for $\delta = d_H$. Assume the right side of the lemma statement holds. First of all, as we have $\max_{\pi \in \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_H(\pi, st) \leq \varepsilon$, we know that for all $i \in \{2, \dots, n-1\}$, we have $d(c_i, st) + r_i \leq \varepsilon$, or $d(c_i, st) \leq \varepsilon - r_i$, so st stabs each disk $D(c_i, \varepsilon - r_i)$ (see Lemma 18 in Appendix A.1). We can draw a similar conclusion for uv . Therefore, each disk $D(c_i, \varepsilon - r_i)$ crosses the entire strip bounded by the tangent lines, with the intersection points splitting it into the left and the right circular arcs. We can thus apply Definition 6 to these arcs, as stated in the lemma.

First suppose that the disks U_1 and U_n do not intersect. Then for any line segment from U_1 to U_n and any disk $D' := D(c_i, \varepsilon - r_i)$, we exit D' after exiting U_1 and enter D' before entering U_n . Hence, for any line pq with $p \in U_1$ and $q \in U_n$ and any $i \in \{2, \dots, n-1\}$, we can find a point $w \in pq \cap D'$; this means that indeed $\max_{w' \in U_i} d(w', pq) \leq \varepsilon$ (see Lemma 22 in Appendix A.2). As this holds for all disks and any choice of p and q , we conclude that $\max_{\pi \in \mathcal{U}} d_H(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.

Now assume that the disks U_1 and U_n intersect. If we consider the line segments pq with $p \in U_1$, $q \in U_n$, we end up in the previous case if either $p \notin U_1 \cap U_n$ or $q \notin U_1 \cap U_n$. So assume that the segment pq lies entirely in the intersection $U_1 \cap U_n$. However, it can be seen that for each disk $D' := D(c_i, \varepsilon - r_i)$, the left boundary of the intersection is to the right of the left boundary of the disk, and the right boundary of the intersection is to the left of the right boundary of the disk; hence, $pq \subset U_1 \cap U_n \subseteq D'$. Therefore, we have $\max_{w' \in U_i} d(w', pq) \leq \varepsilon$, and so also in this case $\max_{\pi \in \mathcal{U}} d_H(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.

We now assume that the right side of the lemma statement is false and show that then $\max_{\pi \in \mathcal{U}} d_H(\pi, \langle \pi(1), \pi(n) \rangle) > \varepsilon$. If $\max_{\pi \in \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_H(\pi, st) > \varepsilon$, then immediately $\max_{\pi \in \mathcal{U}} d_H(\pi, \langle \pi(1), \pi(n) \rangle) > \varepsilon$. Same holds for uv . So, assume those statements hold; then it must be that for at least one intermediate disk the arcs do not lie to the left or to the



■ **Figure 4** Having established the alignments along the two tangents, we can connect them to create a sequence of paths.

right of the arcs of the respective disks. Assume this is disk i , so the disk $D' := D(c_i, \varepsilon - r_i)$. W.l.o.g. assume that the right arc of the disk does not lie entirely to the right of the right arc of U_1 . The argument for the left arc w.r.t. U_n is symmetric.

There must be at least one point p' on the right arc of U_1 that lies outside of D' . Assume for now that U_1 and U_n are disjoint. Then a line segment $p'q$ for any $q \in U_n$ does not stab D' , so $\max_{w' \in U_1} d(w', pq) > \varepsilon$, and so $\max_{\pi \in \mathcal{U}} d_H(\pi, \langle \pi(1), \pi(n) \rangle) > \varepsilon$. If U_1 and U_n intersect, then either p' is outside of the intersection and of D' and there is a point $q \in U_n$ such that $p'q$ does not stab D' ; or we can pick the degenerate line segment $p'p'$, as $p' \in U_1 \cap U_n$, and so $p'p'$ also does not stab D' . In either case, we conclude that $\max_{\pi \in \mathcal{U}} d_H(\pi, \langle \pi(1), \pi(n) \rangle) > \varepsilon$.

For $\delta = d_F$, first assume that $\max_{\pi \in \mathcal{U}} d_F(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$. As $d_F(\pi, \sigma) \geq d_H(\pi, \sigma)$ for any curves π, σ , this also means that $\max_{\pi \in \mathcal{U}} d_H(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$. Furthermore, immediately we get that $\max_{\pi \in \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_F(\pi, st) \leq \varepsilon$, and the same for uv , which yields the right side of the lemma as shown above.

Now assume that the right side holds. As for the Hausdorff distance, we know that the disks cross the entire strip and that Definition 6 applies. It remains to show that for any line segment pq with $p \in U_1, q \in U_n$, there is a valid alignment that maintains the correct ordering and bottleneck distance, assuming it exists for every realisation for st and uv . Consider a valid alignment established for st and uv , so the sequence of points a_i on st and b_i on uv that are mapped to U_i . We can always find such points for each individual U_i (see Lemma 21 in Appendix A.2), and as we know that the Fréchet distance is below the threshold for st and uv , there is such a valid alignment, i.e. we know that $a_i \preceq a_{i+1}$ and $b_i \preceq b_{i+1}$ for all $i \in [n-1]$.

For the rest of the proof, the rough idea is as follows. We can create paths from a_i to b_i so that every segment pq with $p \in U_1$ and $q \in U_n$ crosses these paths in the correct order, thus proving that a Fréchet alignment exists. When U_1 and U_n are disjoint, these paths are simply geodesic paths within the region bounded by the two tangents and the U_1 and U_n . If they intersect, we can instead create these paths by connecting a_i to the top intersection point of the disks and b_i to the bottom intersection point, as in Figure 4. Note that when the two disks intersect and the segment pq goes through the intersection, it may not cross the paths at all; however, every point in the intersection is close enough to all intermediate disks. We now discuss this idea in more detail.

First suppose that the disks U_1 and U_n do not intersect. Consider the region R bounded by the outer tangents and the disk arcs that are not part of the convex hull of $U_1 \cup U_n$. We connect, for each $i \in \{2, \dots, n-1\}$, a_i to b_i with a geodesic shortest path in R , as in Figure 4. We claim that for any line segment pq defined above, the intersection points of the shortest paths with the segment give a valid alignment, yielding $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_F(\pi, pq) \leq \varepsilon$. As the choice of pq was arbitrary, this will complete the proof.

To show that the alignment is valid, we need to show that the order is correct and that the distances fall below the threshold. First consider the case where the geodesic shortest path for point i does not touch the boundary formed by arcs of region R . In this case, it is simply a line segment $a_i b_i$. Note that by definition $a_i, b_i \in D(c_i, \varepsilon - r_i)$; as disks are convex, also $a_i b_i \subset D(c_i, \varepsilon - r_i)$; thus, the intersection point p'_i of pq with $a_i b_i$ is in $D(c_i, \varepsilon - r_i)$, so by Lemma 21, $\max_{w \in U_i} \|p'_i - w\| \leq \varepsilon$. Furthermore, note that $a_i \preceq a_{i+1}$ and $b_i \preceq b_{i+1}$; thus, the line segments $a_i b_i$ and $a_{i+1} b_{i+1}$ cannot cross, so also $p'_i \preceq p'_{i+1}$.

Now w.l.o.g. consider the case where the geodesic shortest path for point i touches the arc of U_1 . The geodesic shortest paths do not cross: on the path from a_i (or b_i) to the arc they form a tangent to the arc, thus for $a_i \preceq a_{i+1}$ the tangent point for a_i comes before that of a_{i+1} when going along the arc from s to u . So, just as in the previous case, these line segments cannot cross. Having reached the arc, both shortest paths will follow it, as otherwise the path would not be a shortest path; thus, the arcs do not cross, either. Finally, a path from the previous case does not touch any path that touches the arc boundary of R by definition. Finally, note that the condition that we have established on the right arcs of disks being to the right of the right arc of U_1 (and symmetric for the left arcs and U_n) means that the geodesic shortest paths that touch the arc boundary of R stay within the respective disks $D(c_i, \varepsilon - r_i)$. Thus, we have established that for all i we have $p'_i \preceq p'_{i+1}$ and $\max_{w \in U_i} \|p'_i - w\| \leq \varepsilon$, concluding the proof for disjoint U_1 and U_n .

Finally, consider the case where U_1 intersects U_n . Above we used geodesic paths within the region R . However, when U_1 intersects U_n , R consists of two disconnected regions. Observe that one region contains a_i and the other contains b_i . To connect a_i with b_i we use the geodesic from a_i to the intersection point of the two inner boundaries of U_1 and U_n that is in the same region of R , the geodesic from b_i to the other intersection point of the inner boundaries, and join these two by a line segment between the intersection points. Any line segment from a point in U_1 to a point in U_n crosses these paths in order, just like in the previous case. If the line segment goes through the intersection, note that any point in the intersection is close enough to all the intermediate objects, as the intersection is the subset of each disk. So, any point in the intersection can be chosen to establish the trivially in-order alignment to all the intermediate objects. ◀

It is worth noting that the case of $U_1 = U_n$ is similar to how we treat the intersection $U_1 \cap U_n$; however, our definition for the ordering between two disks does not apply. So, if $U_1 = U_n$, then $\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$ if and only if $U_1 \subseteq D(c_i, \varepsilon - r_i)$ for all $i \in \{2, \dots, n-1\}$.

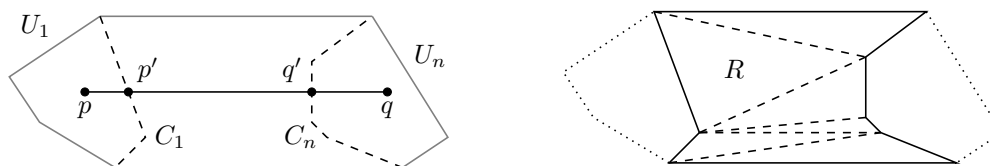
5.2 Non-intersecting PCCSs

Suppose the regions are modelled by convex polygons. Consider first the case where the interiors of U_1 and U_n do not intersect, so at most they share a boundary segment.

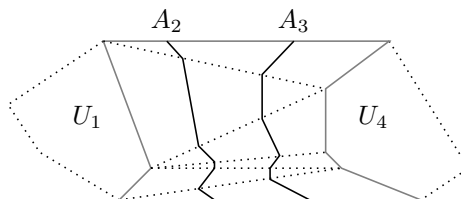
► **Observation 8.** *Given an uncertain curve modelled by convex polygons $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with the interiors of U_1 and U_n not intersecting, note:*

- *There are two outer tangents to the polygons U_1 and U_n , and the convex hull of $U_1 \cup U_n$ consists of a convex chain from U_1 , a convex chain from U_n , and the outer tangents.*
- *Let C_i be the convex chain from U_i that is not part of the convex hull for $i \in \{1, n\}$. Then for $\delta := d_H$ or $\delta := d_F$,*

$$\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon \iff \max_{\pi \in \mathcal{U}, \pi(1) \in C_1, \pi(n) \in C_n} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon.$$



■ **Figure 5** Left: Illustration for Observation 8. The convex hull of the polygons is shown in grey. The dotted chains are C_1 and C_n . Any line segment pq with $p \in U_1$ and $q \in U_n$ crosses C_1 and C_n . Right: Illustration for the procedure. The region R is triangulated.



■ **Figure 6** An example set of curves $A = \{A_2, A_3\}$ discussed in Lemmas 9 and 10.

To see that the second observation is true, note that one direction is trivial. In the other direction, note that any line segment pq with $p \in U_1$, $q \in U_n$ crosses both C_1 and C_n , say, at $p' \in C_1$ and $q' \in C_n$. We know that there is a valid alignment for $p'q'$, both for the Hausdorff and the Fréchet distance; we can then use this alignment for pq . See Figure 5.

We claim that we can check $\max_{\pi \in \mathcal{U}} d_H(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$ using the following procedure (see Figure 5).

1. Triangulate the region R bounded by two convex chains C_1 and C_n and the outer tangents.
2. For each line segment st of the triangulation with $s \in C_1$, $t \in C_n$, and for either $\delta := d_H$ or $\delta := d_F$, check that $\max_{\pi \in \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} \delta(\pi, st) \leq \varepsilon$.

First of all, observe that we can compute a triangulation, and that every triangle has two points from one convex chain and one point from the other chain. If all three points were from the same chain, then the triangle would lie outside of R . Now consider some line segment pq with $p \in C_1$, $q \in C_n$. To complete the argument, it remains to show that the checks in step 2 mean that also $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} \delta(\pi, pq) \leq \varepsilon$. Observe that the triangles span across the region R , so when going from one tangent to the other within R we cross all the triangles. Therefore, we can number the edges of the triangles that go from C_1 to C_n , in the order of occurrence on such a path, from 1 to k . Denote the alignment established on line $j \in [k]$ with the sequence of a_i^j , for $i \in \{2, \dots, n-1\}$; this alignment can be established both for $\delta := d_H$ and $\delta := d_F$. We can establish polygonal curves $A_i := \langle a_i^1, \dots, a_i^k \rangle$; they all stay within R (see Figure 6). We claim that for any line segment pq defined above, we can establish a valid alignment from intersection points of pq and A_i . We do this separately for the Fréchet and the Hausdorff distance.

► **Lemma 9.** *Given a set of curves $A := \{A_2, \dots, A_{n-1}\}$ in R described above for $\delta := d_H$ and a line segment pq with $p \in C_1$, $q \in C_n$, we have $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_H(\pi, pq) \leq \varepsilon$.*

Proof. Note that pq crosses each A_i at least once. We can take any one crossing for each i and establish the alignment. Consider such a crossing point p'_i . It falls in some triangle bounded by a segment from either C_1 or C_n and two line segments that contain points a_i^j and a_i^{j+1} for some $j \in [k-1]$. We know, using Lemma 19, that $\max_{w \in U_i} \|a_i^j - w\| \leq \varepsilon$ and $\max_{w \in U_i} \|a_i^{j+1} - w\| \leq \varepsilon$. Consider any point $w' \in U_i$. Then, using Lemma 14 with $c := d := w'$, we find that $\|w' - p'_i\| \leq \varepsilon$. Therefore, also $\max_{w \in U_i} \|p'_i - w\| \leq \varepsilon$; using Lemma 19, we conclude that indeed $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_H(\pi, pq) \leq \varepsilon$. ◀

26:12 Uncertain Curve Simplification

For the Fréchet distance, we can use the same argument to show closeness; however, we need more care to establish the correct order for the alignment.

► **Lemma 10.** *Given a set of curves $A := \{A_2, \dots, A_{n-1}\}$ in R described above for $\delta := d_F$ and a line segment pq with $p \in C_1$, $q \in C_n$, we have $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_F(\pi, pq) \leq \varepsilon$.*

Proof. Compared to Lemma 9, instead of taking any intersection point of pq with each A_i , we take the *last* intersection point. First, we need to show that curves A_i and A_{i+1} do not cross for any $i \in [n-1]$. Each curve A_i crosses each triangle once, so it suffices to show that a segment $a_i^j a_i^{j+1}$ does not cross $a_{i+1}^j a_{i+1}^{j+1}$. Indeed, as $a_i^j \preceq a_{i+1}^j$ and $a_i^{j+1} \preceq a_{i+1}^{j+1}$, these line segments cannot cross.

Now consider, for each $i \in \{2, \dots, n-1\}$, the polygon P_i bounded by C_1 , A_i , and the corresponding segments of the outer tangents. With the previous statement, it is easy to see that $P_2 \subseteq P_3 \subseteq \dots \subseteq P_{n-1}$. Assume this is not the case, so some $P_i \not\subseteq P_{i+1}$. Then there is a point $z \in P_i$, but $z \notin P_{i+1}$. The point z falls into some triangle with lines j and $j+1$. In this triangle, it means that z is between C_1 and $a_i^j a_i^{j+1}$, but not between C_1 and $a_{i+1}^j a_{i+1}^{j+1}$. However, as these segments do not cross, this would imply that $a_{i+1}^j \prec a_i^j$, but then the check in step 2 would not pass for line j .

Consider the points at which the line segment pq leaves the polygons P_i for the last time. From the definition it is obvious that $p \in P_i$ for all $i \in \{2, \dots, n-1\}$, so this is well-defined. Clearly, due to the subset relationship, the order of such points p'_i is correct, i.e. $p'_i \preceq p'_{i+1}$. Furthermore, each such $p'_i \in A_i$, so using the arguments of Lemma 9 we can show that also the distances are below ε . Thus, we conclude that indeed $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_F(\pi, pq) \leq \varepsilon$. ◀

The proofs of Lemmas 9 and 10 show us how to solve the problem for two convex polygons with non-intersecting interiors. We can also use them directly for the case of line segments that do not intersect except at endpoints.

► **Lemma 11.** *Given $n \in \mathbb{N}^{\geq 3}$, for any imprecise curve modelled with line segments $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = p_i^1 p_i^2 \subset \mathbb{R}^2$ for all $i \in [n]$, given a threshold $\varepsilon \in \mathbb{R}^{>0}$, and given that $U_1 \cap U_n \subset \{p_1^1, p_1^2\}$, and assuming that the triangles $p_1^1 p_n^1 p_1^2$ and $p_1^2 p_n^2 p_1^1$ form a triangulation of the convex hull of $U_1 \cup U_n$, we have $\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$ if and only if*

$$\max \left\{ \begin{array}{l} \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1^1, \pi(n) \equiv p_n^1} \delta(\pi, p_1^1 p_n^1), \quad \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1^2, \pi(n) \equiv p_n^2} \delta(\pi, p_1^2 p_n^2), \\ \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1^2, \pi(n) \equiv p_n^1} \delta(\pi, p_1^2 p_n^1) \end{array} \right\} \leq \varepsilon.$$

We should note that in this particular case it is not necessary to use a triangulation, so we can get rid of the second term; also in the previous proofs a convex partition could work instead, but a triangulation is easier to define.

5.3 Intersecting PCCSs

We now discuss the situation where the interiors of U_1 and U_n intersect, or where line segments U_1 and U_n cross. The argument is the same for $\delta := d_H$ and $\delta := d_F$.

Line segments. Assume line segments $U_1 \stackrel{\text{def}}{=} p_1^1 p_1^2$ and $U_n \stackrel{\text{def}}{=} p_n^1 p_n^2$ cross; call their intersection point s . Then we can use Lemma 11 separately on pairs of $\{p_1^1 s, s p_1^2\} \times \{p_n^1 s, s p_n^2\}$. These pairs cover the entire set of realisations pq with $p \in U_1$, $q \in U_n$, completing the checks.

► **Lemma 12.** *Given $n \in \mathbb{N}^{\geq 3}$, for any imprecise curve modelled with line segments $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = p_i^1 p_i^2 \subset \mathbb{R}^2$ for all $i \in [n]$, given a threshold $\varepsilon \in \mathbb{R}^{>0}$, we can check for both $\delta := d_H$ and $\delta := d_F$, using procedures above, that $\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.*

Convex polygons. Convex polygons whose interiors intersect can be partitioned along the intersection lines, so into a convex polygon $R := U_1 \cap U_n$ and two sets of polygons $\mathcal{P}_1 := \{P_1^1, \dots, P_1^k\}$ and $\mathcal{P}_n := \{P_n^1, \dots, P_n^\ell\}$ for some $k, \ell \in \mathbb{N}^{>0}$. Just as for line segments, we can look at pairs from $\mathcal{P}_1 \times \mathcal{P}_n$ separately. The pairs where R is involved are treated later. Consider some $(P, Q) \in \mathcal{P}_1 \times \mathcal{P}_n$. Note that P and Q are convex polygons with a convex cut-out, so the boundary forms a convex chain, followed by a concave chain. We need to compute some convex polygons P' and Q' with non-intersecting interiors that are equivalent to P and Q , so that we can apply the approaches from Section 5.2.

We claim that we can simply take the convex hull of P and Q to obtain P' and Q' . Clearly, the resulting polygons will be convex. Also, the concave chains of P are bounded by points s and t and are replaced with the line segment st ; same happens for Q with point u and v . The points s, t, u, v are points of intersection of original polygons U_1 and U_n , so they lie on the boundary of R , and their order along that boundary can only be s, t, u, v or s, t, v, u . Thus, it cannot happen that st crosses uv , and it cannot be that uv is in the interior of the convex hull of P , as otherwise R would not be convex. Hence, the interiors of P' and Q' cannot intersect, so they satisfy the necessary conditions.

Finally, we need to show that the solution for (P', Q') is equivalent to that for (P, Q) . One direction is trivial, as $P \subseteq P'$ and $Q \subseteq Q'$; for the other direction, consider any line segment that leaves P through the concave chain. In our approach, we test the lines starting in s and t ; the established alignments are connected into paths. The paths A_i do not cross st . So, any alignment in the region of $\text{CH}(P \cup Q) \setminus (P \cup Q)$ can also be made in the region $\text{CH}(P' \cup Q') \setminus (P' \cup Q')$. So, this approach yields valid solutions for all pairs not involving R .

Now consider the pair (R, R) . A curve may now consist of a single point, so the approach for the Fréchet and the Hausdorff distance is the same: all the points of U_i need to be close enough to all the points of R . To check that, observe that the pair of points $p \in U_i$ and $q \in R$ that has maximal distance has the property that p is an extreme point of U_i in direction qp and q is an extreme point of R in direction pq . So, it suffices, starting at the rightmost point of U_i and leftmost point of R in some coordinate system, to then rotate clockwise around both regions keeping track of the distance between tangent points. Note that only vertices need to be considered, as the extremal point cannot lie on an edge.

Finally, any other pair that involves R is covered by the stronger case of (R, R) : for any line we can align every intermediate object with any point in R . To elaborate, the cases above are not truly a case distinction, as *all* of these combinations should hold; so given a line segment for a pair (P, R) or (R, Q) for some $P \in \mathcal{P}_1$, $Q \in \mathcal{P}_n$, we can pick any point of the segment that lies inside R to establish the alignment, deferring to the stronger previous case (R, R) . Also observe that some line segments covered by the case (P, Q) with $P \in \mathcal{P}_1$, $Q \in \mathcal{P}_n$ may go through R ; this does not impose any unnecessary constraints, so it does not matter that the cases can overlap.

► **Lemma 13.** *Given $n \in \mathbb{N}^{\geq 3}$, for any imprecise curve modelled with convex polygons $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i \subset \mathbb{R}^2$ for all $i \in [n]$ and $V(U_i) = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$, $k \in \mathbb{N}^{>0}$, given a threshold $\varepsilon \in \mathbb{R}^{>0}$, we can check for $\delta := d_H$ and $\delta := d_F$, using procedures above, that $\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.*

6 Combining Steps

In the previous sections, we have shown how to check if a shortcut of length $n \geq 3$ is valid under the Hausdorff or the Fréchet distance, for indecisive points, disks, line segments, and convex polygons. It is easy to see that a shortcut of length $n = 2$ is always valid. Therefore,

■ **Table 1** Running time of our approach in each setting. For indecisive points, k is the number of options per point. For convex polygons, k is the number of vertices.

	Indecisive	Disks	Line segments	Convex polygons
Hausdorff distance	$\mathcal{O}(n^3k^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3k^3)$
Fréchet distance	$\mathcal{O}(n^3k^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3k^3)$

we can use the previously described procedures to construct a shortcut graph; any path in such a graph from the vertex 1 to vertex n corresponds to a valid simplification, so the shortest path gives us the result we need.

► **Theorem 1.** *We can find the shortest vertex-constrained simplification of an uncertain curve, such that for any realisation the simplification is valid, both for the Hausdorff and the Fréchet distance, in time $\mathcal{O}(n^3)$ for uncertainty modelled with disks and line segments, and in time $\mathcal{O}(n^3k^3)$ for uncertainty modelled with indecisive points and convex polygons, where k is the number of options or vertices and n is the length of the curve.*

Proof. Correctness of the approaches has been shown before. We now analyse the running time, also shown in Table 1. For the running time, observe that we need $\mathcal{O}(n^2T)$ time in any setting, due to the shortcut graph construction.

For indecisive points, when testing a shortcut we do $\mathcal{O}(nk)$ -time testing for $\mathcal{O}(k^2)$ combinations of starting and ending points, where k is the number of options per point. For disks, we do a linear number of constant-time checks and two linear-time checks, getting $T \in \mathcal{O}(n)$. For line segments, we also do two (three) linear-time checks per part; two line segments can be split into at most two parts each, so we repeat the process four times. Either way, we get $T \in \mathcal{O}(n)$.

Finally, for convex polygons, assume the complexity of each polygon is at most k . Assume the partitioning resulting from two intersecting polygons yields ℓ_1 and ℓ_2 parts for the first and the second polygon, respectively. Denote the two polygons P and Q and the resulting parts with P_1, \dots, P_{ℓ_1} and Q_1, \dots, Q_{ℓ_2} , respectively. Suppose part P_i has complexity k_i and part Q_j has complexity k'_j , so $|V(P_i)| = k_i$ and $|V(Q_j)| = k'_j$ for some $i \in [\ell_1]$, $j \in [\ell_2]$. We know that every vertex of the original polygons occurs in a constant number of parts, so $\sum_{i=1}^{\ell_1} k_i \in \mathcal{O}(k)$ and $\sum_{j=1}^{\ell_2} k'_j \in \mathcal{O}(k)$; we also know $\ell_1 + \ell_2 \in \mathcal{O}(k)$. We consider all pairs from P and Q , and for each pair we triangulate and do the checks on the triangulation. The triangulation can be done in time $\mathcal{O}((k_i + k'_j) \cdot \log(k_i + k'_j))$, yielding $\mathcal{O}(k_i + k'_j)$ lines, each of which is tested in time $\mathcal{O}(nk)$. The testing dominates, so we need $\mathcal{O}((k_i + k'_j) \cdot nk)$ time. We are interested in

$$\sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \mathcal{O}((k_i + k'_j) \cdot nk) = \mathcal{O}(nk) \cdot \sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \mathcal{O}(k_i + k'_j) = \mathcal{O}(nk^3).$$

So, $T \in \mathcal{O}(nk^3)$ both for the Fréchet and the Hausdorff distance. ◀

7 Conclusion

We have presented approaches for finding the optimal simplification of an uncertain curve under various uncertainty models for the Hausdorff and the Fréchet distance. To recap, we can use Lemmas 7, 12, and 13 and the procedure for indecisive points to test a single shortcut. Constructing a shortcut graph yields the solution. In future work, it would be

interesting to see, similarly to the precise simplification approaches, if an improvement in the running time is possible to subcubic time, or whether one can show a conditional lower bound [8]. It would also be interesting to consider what uncertainty means in the context of global simplification; our approach does not seem easily transferable.

References

- 1 Pankaj K. Agarwal, Boris Aronov, Sariel Har-Peled, Jeff M. Phillips, Ke Yi, and Wuzhou Zhang. Nearest-neighbor searching under uncertainty II. *ACM Transactions on Algorithms*, 13(1):3:1–3:25, 2016. doi:10.1145/2955098.
- 2 Pankaj K. Agarwal, Alon Efrat, Swaminathan Sankararaman, and Wuzhou Zhang. Nearest-neighbor searching under uncertainty I. *Discrete & Computational Geometry*, 58(3):705–745, 2017. doi:10.1007/s00454-017-9903-x.
- 3 Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3):203–219, 2005. doi:10.1007/s00453-005-1165-y.
- 4 Pankaj K. Agarwal and Kasturi R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete & Computational Geometry*, 23(2):273–291, 2000. doi:10.1007/PL00009500.
- 5 Hee-Kap Ahn, Christian Knauer, Marc Scherfenberg, Lena Schlipf, and Antoine Vigneron. Computing the discrete Fréchet distance with imprecise input. *International Journal of Computational Geometry & Applications*, 22(01):27–44, 2012. doi:10.1142/S0218195912600023.
- 6 Sander P. A. Alewijnse, Kevin Buchin, Maike Buchin, Stef Sijben, and Michel A. Westenberg. Model-based segmentation and classification of trajectories. *Algorithmica*, 80(8):2422–2452, 2018. doi:10.1007/s00453-017-0329-x.
- 7 Gill Barequet, Danny Z. Chen, Ovidiu Daescu, Michael T. Goodrich, and Jack S. Snoeyink. Efficiently approximating polygonal paths in three and higher dimensions. *Algorithmica*, 33(2):150–167, 2002. doi:10.1007/s00453-001-0096-5.
- 8 Karl Bringmann and Bhaskar Ray Chaudhury. Polyline simplification has cubic complexity. In *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2019.18.
- 9 Kevin Buchin, Chenglin Fan, Maarten Löffler, Aleksandr Popov, Benjamin Raichel, and Marcel Roeloffzen. Fréchet distance for uncertain curves. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:20, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.20.
- 10 Kevin Buchin, Maximilian Konzack, and Wim Reddingius. Progressive simplification of polygonal curves. *Computational Geometry: Theory & Applications*, 88:101620:1–101620:18, 2020. doi:10.1016/j.comgeo.2020.101620.
- 11 Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Preprocessing imprecise points for Delaunay triangulation: Simplified and extended. *Algorithmica*, 61(3):674–693, 2011. doi:10.1007/s00453-010-9430-0.
- 12 Kevin Buchin, Maarten Löffler, Aleksandr Popov, and Marcel Roeloffzen. Uncertain curve simplification, 2021. arXiv:2103.09223.
- 13 Kevin Buchin, Stef Sijben, T. Jean Marie Arseneau, and Erik P. Willems. Detecting movement patterns using Brownian bridges. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*, pages 119–128, New York, NY, USA, 2012. ACM. doi:10.1145/2424321.2424338.
- 14 Maike Buchin and Stef Sijben. Discrete Fréchet distance for uncertain points, 2016. Presented at EuroCG 2016, Lugano, Switzerland. URL: http://www.eurocg2016.usi.ch/sites/default/files/paper_72.pdf [cited 2019-07-10].

- 15 Leizhen Cai and Mark Keil. Computing visibility information in an inaccurate simple polygon. *International Journal of Computational Geometry & Applications*, 7:515–538, 1997. doi:10.1142/S0218195997000326.
- 16 W. S. Chan and Francis Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry & Applications*, 6(1):59–77, 1996. doi:10.1142/S0218195996000058.
- 17 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. doi:10.3138/FM57-6770-U75U-7727.
- 18 Anne Driemel, Herman Haverkort, Maarten Löffler, and Rodrigo I. Silveira. Flow computations on imprecise terrains. *Journal of Computational Geometry*, 4(1):38–78, 2013. doi:10.20382/jocg.v4i1a3.
- 19 William Evans, David Kirkpatrick, Maarten Löffler, and Frank Staals. Competitive query strategies for minimising the ply of the potential locations of moving points. In *Proceedings of the 29th Annual Symposium on Computational Geometry (SoCG '13)*, pages 155–164, New York, NY, USA, 2013. ACM. doi:10.1145/2462356.2462395.
- 20 Chris Gray, Frank Kammer, Maarten Löffler, and Rodrigo I. Silveira. Removing local extrema from imprecise terrains. *Computational Geometry: Theory & Applications*, 45(7):334–349, 2012. doi:10.1016/j.comgeo.2012.02.002.
- 21 Joachim Gudmundsson, Jyrki Katajainen, Damian Merrick, Cahya Ong, and Thomas Wolle. Compressing spatio-temporal trajectories. *Computational Geometry: Theory & Applications*, 42(9):825–841, 2009. doi:10.1016/j.comgeo.2009.02.002.
- 22 Leonidas J. Guibas, John E. Hershberger, Joseph S. B. Mitchell, and Jack S. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 3(4):383–415, 1993. doi:10.1142/S0218195993000257.
- 23 Hiroshi Imai and Masao Iri. Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, 36(1):31–41, 1986. doi:10.1016/S0734-189X(86)80027-5.
- 24 Allan Jørgensen, Jeff M. Phillips, and Maarten Löffler. Geometric computations on indecisive points. In *Algorithms and Data Structures (WADS 2011)*, volume 6844 of *Lecture Notes in Computer Science (LNCS)*, pages 536–547, Berlin, Germany, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-22300-6_45.
- 25 Christian Knauer, Maarten Löffler, Marc Scherfenberg, and Thomas Wolle. The directed Hausdorff distance between imprecise point sets. *Theoretical Computer Science*, 412(32):4173–4186, 2011. doi:10.1016/j.tcs.2011.01.039.
- 26 Maarten Löffler. *Data Imprecision in Computational Geometry*. PhD thesis, Universiteit Utrecht, 2009. URL: <https://dspace.library.uu.nl/bitstream/handle/1874/36022/loffler.pdf> [cited 2019-06-15].
- 27 Maarten Löffler and Wolfgang Mulzer. Unions of onions: Preprocessing imprecise points for fast onion decomposition. *Journal of Computational Geometry*, 5(1):1–13, 2014. doi:10.20382/jocg.v5i1a1.
- 28 Maarten Löffler and Jeff M. Phillips. Shape fitting on point sets with probability distributions. In *Algorithms – ESA 2009*, number 5757 in *Lecture Notes in Computer Science (LNCS)*, pages 313–324, Berlin, Germany, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-04128-0_29.
- 29 Maarten Löffler and Jack S. Snoeyink. Delaunay triangulations of imprecise points in linear time after preprocessing. *Computational Geometry: Theory & Applications*, 43(3):234–242, 2010. doi:10.1016/j.comgeo.2008.12.007.
- 30 Maarten Löffler and Marc van Kreveld. Largest and smallest tours and convex hulls for imprecise points. In *Algorithm Theory – SWAT 2006*, volume 4059 of *Lecture Notes in Computer Science (LNCS)*, pages 375–387, Berlin, Germany, 2006. Springer Berlin Heidelberg. doi:10.1007/11785293_35.

- 31 Avraham Melkman and Joseph O'Rourke. On polygonal chain approximation. In Godfried T. Toussaint, editor, *Computational Morphology*, volume 6 of *Machine Intelligence and Pattern Recognition*, pages 87–95. Elsevier Science Publishers, 1988. doi:10.1016/B978-0-444-70467-2.50012-6.
- 32 Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. Probabilistic skylines on uncertain data. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 15–26, Los Angeles, CA, USA, 2007. VLDB Endowment. doi:10.5555/1325851.1325858.
- 33 Aleksandr Popov. Similarity of uncertain trajectories. Master's thesis, Eindhoven University of Technology, 2019. URL: <https://research.tue.nl/en/studentTheses/similarity-of-uncertain-trajectories> [cited 2019-12-18].
- 34 Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972. doi:10.1016/S0146-664X(72)80017-0.
- 35 Subhash Suri, Kevin Verbeek, and Hakan Yildiz. On the most likely convex hull of uncertain points. In *Algorithms – ESA 2013*, volume 8125 of *Lecture Notes in Computer Science (LNCS)*, pages 791–802, Berlin, Germany, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40450-4_67.
- 36 Mees van de Kerkhof, Irina Kostitsyna, Maarten Löffler, Majid Mirzanezhad, and Carola Wenk. Global curve simplification. In *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 67:1–67:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2019.67.
- 37 Marc van Kreveld, Maarten Löffler, and Joseph S. B. Mitchell. Preprocessing imprecise points and splitting triangulations. *SIAM Journal on Computing*, 39(7):2990–3000, 2010. doi:10.1137/090753620.

A Shortcut Testing: Intermediate Points

In this appendix, we discuss testing a single shortcut where we fix the realisations of the first and the last uncertain point. We provided intuitive explanations in Section 4 and discuss the details here. We state some basic facts about the Hausdorff and the Fréchet distance in the precise setting (see the full version [12] for the proofs) and use them to design simple algorithms for testing shortcuts in the uncertain settings.

A.1 Hausdorff Distance

We start by recalling some useful facts about the Hausdorff distance in the precise setting.

► **Lemma 14.** *Given points $a, b, c, d \in \mathbb{R}^2$ forming segments ab and cd , the largest distance from one segment to the other is $\max_{p \in ab} d(p, cd) = \max\{d(a, cd), d(b, cd)\}$.*

► **Lemma 15.** *Given $n \in \mathbb{N}^{>0}$, for any precise curve $\pi = \langle p_1, \dots, p_n \rangle$ with $p_i \in \mathbb{R}^2$ for all $i \in [n]$, we have $d_H(\pi, p_1 p_n) = \max_{i \in [n]} d(p_i, p_1 p_n)$.*

Indecisive points. We generalise the setting to include imprecision. We first claim that the straightforward setting with indecisive points permits an easy solution using Lemma 15; the proof can be found in the full version [12].

► **Lemma 16.** *Given $n, k \in \mathbb{N}^{>0}$, for any indecisive curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $n \geq 3$, $U_i = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$ and $p_i^j \in \mathbb{R}^2$ for all $i \in [n]$, $j \in [k]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_H(\pi, p_1 p_n) = \max_{i \in \{2, \dots, n-1\}} \max_{j \in [k]} d(p_i^j, p_1 p_n)$.*

Note that this means that when the start and end realisations are fixed, we can test that a shortcut is valid using the lemma above in time $\mathcal{O}(nk)$ for a shortcut of length n .

26:18 Uncertain Curve Simplification

Disks. We proceed to present the way to test shortcuts for fixed realisations of the first and the last points when the imprecision is modelled using disks. In the next arguments the following form of a triangle inequality is useful (again, see the full version [12] for details).

► **Lemma 17.** For any $p, q \in \mathbb{R}^2$ and a line segment ab on $a, b \in \mathbb{R}^2$, $d(p, ab) \leq \|p - q\| + d(q, ab)$.

► **Lemma 18.** Given $n \in \mathbb{N}^{\geq 3}$, for any imprecise curve modelled with disks $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = D(c_i, r_i)$ for all $i \in [n]$ and $c_i \in \mathbb{R}^2$, $r_i \in \mathbb{R}^{\geq 0}$ for all $i \in [n]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have

$$\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_H(\pi, p_1 p_n) = \max_{i \in \{2, \dots, n-1\}} (d(c_i, p_1 p_n) + r_i).$$

Proof. Assume the setting of the lemma. We derive

$$\begin{aligned} \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_H(\pi, p_1 p_n) &= \{\text{Lemma 15}\} \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \max_{i \in [n]} d(\pi(i), p_1 p_n) \\ &= \{\text{Def. } \Subset, d(p_1, p_1 p_n) = d(p_n, p_1 p_n) = 0\} \max_{i \in \{2, \dots, n-1\}} \max_{p \in U_i} d(p, p_1 p_n). \end{aligned}$$

It remains to show that $\max_{p \in U_i} d(p, p_1 p_n) = d(c_i, p_1 p_n) + r_i$ for any $i \in \{2, \dots, n-1\}$.

Pick $p' := \operatorname{argmax}_{p \in U_i} d(p, p_1 p_n)$. Note that by Lemma 17, $d(p', p_1 p_n) \leq \|p' - c_i\| + d(c_i, p_1 p_n)$. Furthermore, as $p' \in U_i$, by definition of U_i we have $\|p' - c_i\| \leq r_i$. Thus, $\max_{p \in U_i} d(p, p_1 p_n) \leq d(c_i, p_1 p_n) + r_i$, and we need to show the other direction.

Now pick a point $q' := \operatorname{argmin}_{q \in p_1 p_n} \|q - c_i\|$, so that $d(c_i, p_1 p_n) = \|q' - c_i\|$. Draw the line through c_i and q' and pick the point p' on that line on the boundary of U_i on the opposite side of q w.r.t. c_i . Clearly, $\|p' - c_i\| = r_i$ and $q' = \operatorname{argmin}_{q \in p_1 p_n} \|q - p'\|$. Thus,

$$d(p', p_1 p_n) = \|p' - q'\| = \|q' - c_i\| + \|p' - c_i\| = d(c_i, p_1 p_n) + r_i.$$

Note that $p' \in U_i$, so we conclude $\max_{p \in U_i} d(p, p_1 p_n) \geq d(c_i, p_1 p_n) + r_i$. ◀

This lemma allows us to test a shortcut in time $\mathcal{O}(n)$ for a shortcut of length n .

Polygonal closed convex sets.

► **Lemma 19.** Given $n, k \in \mathbb{N}^{>0}$, $n \geq 3$, for any imprecise curve modelled with PCCSs $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i \subset \mathbb{R}^2$ and $V(U_i) = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have

$$\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_H(\pi, p_1 p_n) = \max_{i \in \{2, \dots, n-1\}} \max_{v \in V(U_i)} d(v, p_1 p_n).$$

Proof. Assume the setting of the lemma. As before, derive

$$\begin{aligned} \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_H(\pi, p_1 p_n) &= \{\text{Lemma 15}\} \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \max_{i \in [n]} d(\pi(i), p_1 p_n) \\ &= \{\text{Def. } \Subset, d(p_1, p_1 p_n) = d(p_n, p_1 p_n) = 0\} \max_{i \in \{2, \dots, n-1\}} \max_{p \in U_i} d(p, p_1 p_n). \end{aligned}$$

To show that the claim holds, it remains to show that for any PCCS U and a line segment ab , $\max_{p \in U} d(p, ab) = \max_{v \in V(U)} d(v, ab)$. Firstly, as $V(U) \subset U$, we immediately have $\max_{p \in U} d(p, ab) \geq \max_{v \in V(U)} d(v, ab)$. Consider any $p \in U$. We show that there is some $v \in V(U)$ such that $d(v, ab) \geq d(p, ab)$, completing the proof, with a case distinction on p .

■ **Algorithm 1** Testing a shortcut on an indecisive curve with the Fréchet distance.

Require: $\mathcal{U} = \langle U_1, \dots, U_n \rangle$, $n, k \in \mathbb{N}^{>0}$, $\forall i \in [n] : U_i = \{p_i^1, \dots, p_i^k\}$, $\forall i \in [n], j \in [k] : p_i^j \in \mathbb{R}^2$, $\varepsilon \in \mathbb{R}^{>0}$, $p_1 \in U_1, p_n \in U_n$

- 1: **function** CHECKFRÉCHETIND($\mathcal{U}, p_1, p_n, n, k, \varepsilon$)
- 2: $s_1 := 1$
- 3: **for** $i \in \{2, \dots, n-1\}$ **do**
- 4: $T_i := \emptyset$
- 5: **for** $j \in [k]$ **do**
- 6: $S_i^j := \{t \in [s_{i-1}, 2] \mid \|p_i^j - p_1 p_n(t)\| \leq \varepsilon\}$
- 7: **if** $S_i^j = \emptyset$ **then**
- 8: **return** False
- 9: $T_i := T_i \cup \min S_i^j$
- 10: $s_i := \max T_i$
- 11: **return** True

- $p \in V(U)$. Then pick $v := p$, and we are done.
- $p \notin V(U)$, but p is on the boundary of U . Consider the vertices $v, w \in V(U)$ with $p \in vw$. Using Lemma 14, we note $\max_{q \in vw} d(q, ab) = \max\{d(v, ab), d(w, ab)\}$. W.l.o.g. suppose $d(v, ab) \geq d(w, ab)$. Then for v indeed we have $d(v, ab) \geq d(p, ab)$.
- p is in the interior of U (cannot occur for line segments). Find the point $q' := \operatorname{argmin}_{q \in ab} \|p - q\|$, so $d(p, ab) = \|p - q'\|$. Draw the line through p and q' ; let p' be the point on that line on the boundary of U on the opposite side of q' w.r.t. p . Clearly, $q' = \operatorname{argmin}_{q \in ab} \|p' - q\|$, so $d(p', ab) > d(p, ab)$. Then we can find a vertex $v \in V(U)$ as in the previous cases, yielding $d(v, ab) \geq d(p', ab) > d(p, ab)$.

This covers all the cases, so the statement holds. ◀

As before, this lemma gives us a simple way to test the shortcut with fixed realisations of the first and the last points in time $\mathcal{O}(nk)$ for a shortcut of length n and PCCSs with k vertices.

A.2 Fréchet Distance

We omit results for the Fréchet distance in the precise setting here; see the full version [12].

Indecisive points. The idea is that in the precise case we can always align greedily as we move along the line segment. In this case, we also need to find the realisation for each indecisive point that makes for the ‘worst’ greedy choice.

► **Lemma 20.** *Given $n, k \in \mathbb{N}^{>0}$ and $\varepsilon \in \mathbb{R}^{>0}$, for any indecisive trajectory $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$ and $p_i^j \in \mathbb{R}^2$ for all $i \in [n], j \in [k]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have, using Algorithm 1,*

$$\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_F(\pi, p_1 p_n) \leq \varepsilon \iff \text{CHECKFRÉCHETIND}(\mathcal{U}, p_1, p_n, n, k, \varepsilon) = \text{True}.$$

Proof. First, assume that $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_F(\pi, p_1 p_n) \leq \varepsilon$. In the algorithm, we compute some set S_i^j for each p_i^j and then pick one value from it and add it to T_i ; from T_i we then pick a single value as s_i . So, $s_i \in S_i^j$ for some $j_i \in [k]$, on every iteration $i \in \{2, \dots, n-1\}$. Consider a realisation $\pi \in \mathcal{U}$ with $\pi(1) \equiv p_1$, $\pi(n) \equiv p_n$, and $\pi(i) \equiv p_i^{j_i}$ for every $i \in \{2, \dots, n-1\}$, where j_i is chosen as the value corresponding to s_i . Then we know $d_F(\pi, p_1 p_n) \leq \varepsilon$. So, there is an alignment that can be given as a sequence of n positions, $t_i \in [1, 2]$, such that $\|\pi(i) - p_1 p_n(t_i)\| \leq \varepsilon$ and $t_i \leq t_{i+1}$ for all i . The alignment is established by interpolating between the consecutive points on the curves (see Section 2).

26:20 Uncertain Curve Simplification

We now show by induction that $s_i \leq t_i$ for all i . For $i = 2$, we get, for the chosen j_2 , $s_2 := \min\{t \in [1, 2] \mid \|p_2^{j_2} - p_1 p_n(t)\| \leq \varepsilon\}$. As we have $t_2 \in \{t \in [1, 2] \mid \|p_2^{j_2} - p_1 p_n(t)\| \leq \varepsilon\}$, we get $s_2 \leq t_2$. Now assume the statement holds for some i , then for $i + 1$ we get $s_{i+1} := \min\{t \in [s_i, 2] \mid \|p_{i+1}^{j_{i+1}} - p_1 p_n(t)\| \leq \varepsilon\}$; we can rephrase this so that

$$s_{i+1} \stackrel{\text{def}}{=} \min(\{t \in [1, 2] \mid \|p_{i+1}^{j_{i+1}} - p_1 p_n(t)\| \leq \varepsilon\} \cap [s_i, 2]).$$

So, there are two options.

- $s_{i+1} = s_i$. Then we know $s_{i+1} = s_i \leq t_i \leq t_{i+1}$.
- $s_{i+1} > s_i$. Then we can use the same argument as for $i = 2$ to find that $s_{i+1} \leq t_{i+1}$.

Now we know that for every i , $t_i \in S_i^{j_i}$ for the choice of j_i described above. Therefore, for any $p_{i+1}^{j_{i+1}}$ there is always a realisation prefix such that any valid alignment has $t_{i+1} \geq s_i$; as we know that there is a valid alignment for every realisation, we conclude that every S_i^j is non-empty. Thus, the algorithm returns True.

Now assume that the algorithm returns True. Consider any realisation $\pi \in \mathcal{U}$. We claim that there is a valid alignment, described with a sequence of $t_i \in [1, 2]$ for $i \in \{2, \dots, n-1\}$, such that $s_{i-1} \leq t_i \leq s_i$ and $\|p_1 p_n(t_i) - \pi(i)\| \leq \varepsilon$. Denote the realisation $\pi \stackrel{\text{def}}{=} \langle p_1, p_2^{j_2}, p_3^{j_3}, \dots, p_{n-1}^{j_{n-1}}, p_n \rangle$, so the sequence $\langle j_2, \dots, j_{n-1} \rangle$ describes the choices of the realisation. Consider the set $S_i^{j_i}$ for any $i \in \{2, \dots, n-1\}$. We know that it is non-empty, otherwise the algorithm would have returned False. We claim that we can pick $t_i = \min S_i^{j_i}$ for every i . By definition, $S_i^{j_i} \subseteq [1, 2]$ and $\|p_1 p_n(t_i) - \pi(i)\| \leq \varepsilon$. We also trivially get that $s_{i-1} \leq t_i$. Finally, note that $t_i \in T_i$, and $s_i := \max T_i$, so $t_i \leq s_i$.

This argument shows that $t_i \leq t_{i+1}$ for every i , and that $\|p_1 p_n(t_i) - \pi(i)\| \leq \varepsilon$. Therefore, $d_F(\pi, p_1 p_n) \leq \varepsilon$. As this works for any realisation with $\pi(1) \equiv p_1$ and $\pi(n) \equiv p_n$, we conclude $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_F(\pi, p_1 p_n) \leq \varepsilon$. ◀

Disks. To show the generalisation to disks, it is helpful to reframe the problem as that of disk stabbing for appropriate disks. We state some useful facts first (see the full version [12]).

► **Lemma 21.** *Given a disk $D_1 := D(c, r)$ with $c \in \mathbb{R}^2$, $r \in \mathbb{R}^{\geq 0}$, a threshold $\varepsilon \in \mathbb{R}^{> 0}$, and a point $p \in \mathbb{R}^2$, define $D_2 := D(c, \varepsilon - r)$. We have $\max_{p' \in D_1} \|p - p'\| \leq \varepsilon \iff p \in D_2$.*

► **Lemma 22.** *Given a disk $D_1 := D(c, r)$ with $c \in \mathbb{R}^2$, $r \in \mathbb{R}^{\geq 0}$, $\varepsilon \in \mathbb{R}^{> 0}$, and a line segment pq with $p, q \in \mathbb{R}^2$, define $D_2 := D(c, \varepsilon - r)$. Then $\max_{p' \in D_1} d(p', pq) \leq \varepsilon \iff pq \cap D_2 \neq \emptyset$.*

■ **Algorithm 2** Testing a shortcut on an imprecise curve modelled with disks with the Fréchet distance.

Require: $\mathcal{U} = \langle U_1, \dots, U_n \rangle$, $n \in \mathbb{N}^{> 0}$, $\forall i \in [n] : U_i = D(c_i, r_i)$, $\forall i \in [n] : c_i \in \mathbb{R}^2, r_i \in \mathbb{R}^{\geq 0}$, $\varepsilon \in \mathbb{R}^{> 0}$, $p_1 \in U_1$, $p_n \in U_n$

```

1: function CHECKFRÉCHETDISKS( $\mathcal{U}, p_1, p_n, n, \varepsilon$ )
2:    $s_1 := 1$ 
3:   for  $i \in \{2, \dots, n-1\}$  do
4:      $S_i := \{t \in [s_{i-1}, 2] \mid \|c_i - p_1 p_n(t)\| \leq \varepsilon - r_i\}$ 
5:     if  $S_i = \emptyset$  then
6:       return False
7:      $s_i := \min S_i$ 
8:   return True

```

► **Lemma 23.** *Given $n \in \mathbb{N}^{>0}$ and $\varepsilon \in \mathbb{R}^{>0}$, for any imprecise curve modelled with disks $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = D(c_i, r_i)$ for all $i \in [n]$ and $c_i \in \mathbb{R}^2$, $r_i \in \mathbb{R}^{\geq 0}$ for all $i \in [n]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have, using Algorithm 2,*

$$\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_F(\pi, p_1 p_n) \leq \varepsilon \iff \text{CHECKFRÉCHETDISKS}(\mathcal{U}, p_1, p_n, n, \varepsilon) = \text{True}.$$

Proof. We use Lemma 22 to change the problem: rather than establishing an alignment that comes in the correct order and satisfies the distance constraints, we do disk stabbing and pick the stabbing points in the correct order. So, we have $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_F(\pi, p_1 p_n) \leq \varepsilon$ if and only if there exists a sequence of points $p'_i \in p_1 p_n \cap D(c_i, \varepsilon - r_i)$ for all $i \in \{2, \dots, n-1\}$ such that $p'_i \preceq p'_{i+1}$ along $p_1 p_n$ for all $i \in \{2, \dots, n-2\}$. We show that this is exactly what Algorithm 2 computes in the full version [12]. ◀

■ **Algorithm 3** Testing a shortcut on an imprecise curve modelled with PCCSs with the Fréchet distance.

Require: $\mathcal{U} = \langle U_1, \dots, U_n \rangle$, $n, k \in \mathbb{N}^{>0}$, $\forall i \in [n] : U_i$ is a PCCS, $V(U_i) = \{p_i^1, \dots, p_i^k\}$, $\forall i \in [n], j \in [k] : p_i^j \in \mathbb{R}^2$, $\varepsilon \in \mathbb{R}^{>0}$, $p_1 \in U_1$, $p_n \in U_n$

- 1: **function** CHECKFRÉCHETPCCS($\mathcal{U}, p_1, p_n, n, k, \varepsilon$)
- 2: $s_1 := 1$
- 3: **for** $i \in \{2, \dots, n-1\}$ **do**
- 4: $T_i := \emptyset$
- 5: **for** $j \in [k]$ **do**
- 6: $S_i^j := \{t \in [s_{i-1}, 2] \mid \|p_i^j - p_1 p_n(t)\| \leq \varepsilon\}$
- 7: **if** $S_i^j = \emptyset$ **then**
- 8: **return** False
- 9: $T_i := T_i \cup \min S_i^j$
- 10: $s_i := \max T_i$
- 11: **return** True

Polygonal closed convex sets.

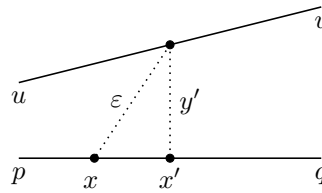
► **Lemma 24.** *Given $n, k \in \mathbb{N}^{>0}$ and $\varepsilon \in \mathbb{R}^{>0}$, for any imprecise curve modelled with PCCSs $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i \subset \mathbb{R}^2$ and $V(U_i) = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have, using Algorithm 3,*

$$\max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_F(\pi, p_1 p_n) \leq \varepsilon \iff \text{CHECKFRÉCHETPCCS}(\mathcal{U}, p_1, p_n, n, k, \varepsilon) = \text{True}.$$

Proof. As we have shown in Lemma 19, it suffices to test the vertices of a PCCS to establish that the distance from every point to the line segment is below the threshold. It remains to show that the extreme alignment (in terms of ordering) for the Fréchet distance is also achieved at a vertex. This case then becomes identical to the indecisive points case, so we can use Lemma 20 to show correctness.

Consider an arbitrary point $t \in U_i$ and let s be the earliest point in the ε -disk around t that is on pq . Clearly, if t is in the interior of U_i , then we can take any t' on the line through t parallel to pq and get the corresponding s' with $s \prec s'$. So, assume t is on the boundary of U_i . Suppose that $t \in uv$ with $u, v \in V(U_i)$. Rotate and translate the coordinate plane so that pq lies on the x -axis. Derive the equation for the line containing uv , say, $y' = kx' + b$. First consider $k = 0$, so the line containing uv is parallel to the line containing pq . In this case, clearly, moving along uv in the direction coinciding with the direction from p to q

26:22 Uncertain Curve Simplification



■ **Figure 7** Illustration for the computation in Lemma 24.

increases the x -coordinate of point of interest, so moving to a vertex is optimal. Now assume $k > 0$. If $k < 0$, reflect the coordinate plane about $y = 0$. Geometrically, it is easy to see (Figure 7) that the coordinate of interest can be expressed as

$$x = x' - \sqrt{\varepsilon^2 - y'^2} = \frac{y' - b}{k} - \sqrt{\varepsilon^2 - y'^2}.$$

We want to maximise x by picking the appropriate y' . We take the derivative: $dx/dy' = 1/k + y'/\sqrt{\varepsilon^2 - y'^2}$. We equate it to 0 to find the critical point of the function, $y'_0 = -\varepsilon/\sqrt{k^2 + 1}$. We can check that for $y' < y'_0$, the value of the derivative is negative, and for $y' > y'_0$ it is positive, so at $y' = y'_0$ we achieve a local minimum. There are no other critical points, so to maximise x , we want to move away from the local minimum as far as possible. As we are limited to the line segment uv , the maximum is achieved at an endpoint. ◀

Fractional Homomorphism, Weisfeiler-Leman Invariance, and the Sherali-Adams Hierarchy for the Constraint Satisfaction Problem

Silvia Butti   

Department of Information and Communication Technologies,
Universitat Pompeu Fabra, Barcelona, Spain

Víctor Dalmau   

Department of Information and Communication Technologies,
Universitat Pompeu Fabra, Barcelona, Spain

Abstract

Given a pair of graphs \mathbf{A} and \mathbf{B} , the problems of deciding whether there exists either a homomorphism or an isomorphism from \mathbf{A} to \mathbf{B} have received a lot of attention. While graph homomorphism is known to be NP-complete, the complexity of the graph isomorphism problem is not fully understood. A well-known combinatorial heuristic for graph isomorphism is the Weisfeiler-Leman test together with its higher order variants. On the other hand, both problems can be reformulated as integer programs and various LP methods can be applied to obtain high-quality relaxations that can still be solved efficiently. We study so-called fractional relaxations of these programs in the more general context where \mathbf{A} and \mathbf{B} are not graphs but arbitrary relational structures. We give a combinatorial characterization of the Sherali-Adams hierarchy applied to the homomorphism problem in terms of fractional isomorphism. Collaterally, we also extend a number of known results from graph theory to give a characterization of the notion of fractional isomorphism for relational structures in terms of the Weisfeiler-Leman test, equitable partitions, and counting homomorphisms from trees. As a result, we obtain a description of the families of CSPs that are closed under Weisfeiler-Leman invariance in terms of their polymorphisms as well as decidability by the first level of the Sherali-Adams hierarchy.

2012 ACM Subject Classification Mathematics of computing \rightarrow Graph theory; Theory of computation \rightarrow Constraint and logic programming

Keywords and phrases Weisfeiler-Leman algorithm, Sherali-Adams hierarchy, Graph homomorphism, Constraint Satisfaction Problem

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.27

Related Version *Full Version:* <https://arxiv.org/abs/2107.02956>

Funding *Silvia Butti:* The project that gave rise to these results received the support of a fellowship from “la Caixa” Foundation (ID 100010434). The fellowship code is LCF/BQ/DI18/11660056. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713673.

Víctor Dalmau: Víctor Dalmau was supported by MICCIN grant PID2019-109137GB-C22.

1 Introduction

The graph isomorphism and homomorphism problems, that is, the problems of deciding, given input graphs \mathbf{A} and \mathbf{B} , whether \mathbf{A} is isomorphic (respectively homomorphic) to \mathbf{B} , have been the subject of extensive research. Despite an important research effort, it is still an open problem to determine whether the isomorphism problem for graphs can be solved in polynomial time. The recent quasipolynomial algorithm for the problem presented by Babai [4] is widely regarded as a major breakthrough in theoretical computer science.



© Silvia Butti and Víctor Dalmau;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 27; pp. 27:1–27:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In turn, the complexity of the graph homomorphism problem has also been intensively studied in combinatorics (see [21]) as well as in a more general setting, known as the Constraint Satisfaction Problem (CSP), where \mathbf{A} and \mathbf{B} are not required to be graphs but can be arbitrary relational structures. The CSP is general enough to encompass problems from areas as diverse as artificial intelligence, optimization, computer algebra, computational biology, computational linguistics, among many other. In contrast with the graph isomorphism problem it was quickly established that the homomorphism problem is NP-complete even for graphs, as it can encode graph coloring. Consequently, an important research effort has been put into identifying tractable fragments of the problem, in particular by fixing the target structure \mathbf{B} . This culminated in the recent major result of Bulatov [8] and Zhuk [37], which confirmed a conjecture by Feder and Vardi [15] that, given a fixed target structure \mathbf{B} , the homomorphism problem for \mathbf{B} is either solvable in polynomial time or NP-complete.

Linear programming relaxations, among other relaxations such as SDP-based, have been largely used in the study of both the isomorphism and the homomorphism problem. In fact, the isomorphism problem for graphs \mathbf{A} , \mathbf{B} can be reformulated as an integer program which asks whether there exists a permutation matrix X such that $XN_{\mathbf{A}} = N_{\mathbf{B}}X$, where $N_{\mathbf{A}}$ and $N_{\mathbf{B}}$ are the adjacency matrices of \mathbf{A} , \mathbf{B} respectively. If we relax this condition to only require that X is doubly stochastic, we obtain what is known as fractional isomorphism.

Fractional isomorphism has a combinatorial counterpart in the 1-dimensional Weisfeiler-Leman (1-WL) algorithm [26], also known as colour refinement. In particular, it was shown in [34, 35] and [30] that \mathbf{A} and \mathbf{B} are fractionally isomorphic if and only if 1-WL does not distinguish between them. 1-WL produces a sequence of colourings c_0, c_1, \dots of the nodes of a graph by means of an iterative refinement procedure which assigns a pair of nodes to the same colour class of c^i if they belong to the same class of c^{i-1} , and additionally they have the same number of neighbours of each colour in c^{i-1} . The algorithm keeps iterating until a fixed point is reached. The Weisfeiler-Leman algorithm is a very powerful heuristic to test for graph isomorphism: if two graphs are distinguished by 1-WL (that is, they give rise to distinct fixed-point colourings up to renaming of the vertices), then this is a witness that the graphs are not isomorphic. In fact, it was shown that 1-WL decides the isomorphism problem on almost all graphs (that is, all but $o(2^{\binom{n}{2}})$ graphs on n vertices for every n) [5].

However, it is also easy to see that 1-WL fails on some very simple instances, such as regular graphs. To address these limitations, the original Weisfeiler-Leman algorithm has been extended so that at every iteration on a graph \mathbf{A} it produces a colouring of the set of k -tuples ($k > 1$) of nodes of \mathbf{A} . It was initially conjectured that this hierarchy of increasingly powerful methods, known as the k -dimensional Weisfeiler-Leman (k -WL) algorithm, would provide a polynomial time graph isomorphism test at least for graphs of bounded degree. While this conjecture was proved to be incorrect [10], k -WL turns out to have a number of useful applications, such as the important role it plays in the aforementioned quasipolynomial algorithm of Babai.

In addition, the k -WL algorithm has proven to be very robust and has a number of equivalent formulations. In [14], a new characterization was given in terms of counting homomorphisms: for every $k \geq 1$, \mathbf{A} and \mathbf{B} are indistinguishable by the k -WL algorithm if for every graph \mathbf{T} of treewidth at most k , the number of homomorphisms from \mathbf{T} to \mathbf{A} is equal to the number of homomorphisms from \mathbf{T} to \mathbf{B} . Further, it has been shown in [10] that \mathbf{A} and \mathbf{B} are indistinguishable by the k -WL algorithm if and only if they are indistinguishable in the logic C^{k+1} consisting of all FO formulas with counting quantifiers that have at most $k + 1$ variables.

Similarly to the Weisfeiler-Leman hierarchy for fractional isomorphism, any LP relaxation of an integer 0/1-program, like that of graph isomorphism, can be further strengthened by sequentially applying so-called lift-and-project methods from mathematical programming in order to obtain a hierarchy of increasingly tighter relaxations which can still be solved efficiently. The main idea behind these is to add auxiliary variables and valid inequalities to an initial relaxation of a 0/1 integer program. These methods, which include Lovász-Schrijver [27] and Sherali-Adams [32], have been used to study classical problems in combinatorial optimization such as Max-Cut, Vertex Cover, Maximal Matching, among many others.

Quite surprisingly, Atserias and Maneva [3] and Malkin [28] were able to lift the connection between the 1-WL algorithm and fractional isomorphism to show a close correspondence between the higher levels of the SA hierarchy for the graph isomorphism problem and the k -WL algorithm (and, hence, with the logic C^{k+1}). This correspondence was further tightened in [20].

Let us now turn our attention to the homomorphism problem. Here, LP relaxations have been intensively used in the more general setting of the Constraint Satisfaction Problem and, most usually, in the approximation of its optimization versions such as MaxCSP, consisting of finding a map from the universe of \mathbf{A} to the universe of \mathbf{B} that maximizes the number of constraints satisfied, among other variants. For instance, a simple linear programming relaxation yields a 2-approximation algorithm for the Vertex Cover problem, and no better polynomial time approximation algorithm is known.

One of the simplest and most widespread LP relaxations of the homomorphism problem (see for example [24]) has an algebraic characterization that very much resembles that of fractional isomorphism. For the sake of simplicity we shall present it here in the restricted case of graphs. Let us start with the algebraic formulation of fractional isomorphism which can be, alternatively, expressed as the existence of a pair of doubly stochastic matrices X and Y such that $XM_{\mathbf{A}} = M_{\mathbf{B}}Y$ and $M_{\mathbf{A}}Y^T = X^TM_{\mathbf{B}}$, where $M_{\mathbf{A}}$ denotes the *incidence* matrix of \mathbf{A} . If we relax this condition to only require that X and Y are left stochastic and, additionally, we drop the second equation, then we obtain a relaxation of graph homomorphism, which we call fractional homomorphism. With some minor variations depending on whether the objective function is present (as in MaxCSP) or not and how repeated elements in a tuple are treated, this LP formulation has been extensively used [7, 12, 13, 18, 23, 24].

We consider relaxations arising from the application of the Sherali-Adams (SA) method. Giving an explicit description of the inequalities produced by the SA method for homomorphism might be relatively cumbersome (because the constraints of the input structures \mathbf{A} and \mathbf{B} must be encoded in the polytope-defining inequalities rather than in the objective function as in the optimization variants) even when the target structure \mathbf{B} is fixed. Hence, we consider a simpler family of LP inequalities, interleaved with the SA hierarchy, of which fractional homomorphism corresponds to the first level. Our hierarchy coincides with the usual SA hierarchy for MaxCSP [11, 17, 33, 36] where the objective function has been turned into a constraint.

Our main result is a combinatorial characterization of this family of LP relaxations which, abusing slightly notation, we still shall refer to as $SA^k, k \geq 1$. Along the way, we extend a number of the aforementioned results from graph isomorphism to general isomorphism between relational structures which yields a hierarchy of relaxations of (relational structure) isomorphism. Our results show that, for every $k \geq 1$, the k th level relaxation of the homomorphism problem, that is, SA^k , is tightly related with the corresponding k th level relaxation of isomorphism, which we denote using \equiv_k .

Let us consider the first level of the hierarchy, i.e, fractional homomorphism and fractional isomorphism. In the case of graphs \equiv_1 -equivalence coincides precisely with 1-WL equivalence. Here we show that a structure \mathbf{A} is fractionally homomorphic to a structure \mathbf{B} if and only if there exists a sequence of structures $\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_n$ with $\mathbf{A} = \mathbf{X}_0$, $\mathbf{B} = \mathbf{X}_n$, and for every $i < n$, \mathbf{X}_i is either homomorphic or \equiv_1 -equivalent to \mathbf{X}_{i+1} . The correspondence for any higher level k is obtained by replacing \mathbf{X}_0 and \mathbf{X}_n by suitably defined structures \mathbf{A}_k^* , \mathbf{B}_k^* that allow to reduce the k^{th} level of the hierarchy to the first level.

In particular, it follows that feasibility in SA^k is preserved by structures that are \equiv_k -equivalent.

In essence, this is due to the fact that the LP relaxation of homomorphism inherits the symmetries of \mathbf{A} and \mathbf{B} . There are, indeed, several results exploring the symmetries of an LP program in a similar fashion. In [3] such types of symmetries are used to transfer results between the logic C^k and LP relaxations of several combinatorial problems, whereas [19] aims to identify the partition classes (as in the WL algorithm) of the variables and constraints of an LP program so that, by identifying those in the same class, the LP size is reduced. That being said, the main interest of our result lies precisely in the opposite direction: that is, the fact that the fractional homomorphism LP relaxation is able to certify that \mathbf{A} is not homomorphic to \mathbf{B} unless \mathbf{A} belongs to the backwards closure of \mathbf{B} under homomorphism and \equiv_1 equivalence, or, even more strongly, unless \mathbf{A} is homomorphic to a structure \mathbf{X}_1 which is \equiv_1 -equivalent to a structure \mathbf{X}_2 which in turn is homomorphic to \mathbf{B} .

We apply our results to study the following question: for which structures \mathbf{B} is the set $\text{CSP}(\mathbf{B})$, which contains all the structures homomorphic to \mathbf{B} , closed under 1-WL equivalence. This question arises in the context of solving CSPs in a distributed manner [9] where the elements of the input instance (nodes, edges) are distributed among agents which communicate with each other by sending messages through fixed communication channels. Using our main results we can rederive the classification obtained in [9] using substantially different techniques.

2 Preliminaries

Relational structures

For a positive integer n , We denote by $[n]$ the set $\{1, \dots, n\}$. We shall denote tuples in boldface. Let $\mathbf{a} = (a_1, \dots, a_k) \in A^k$. We use $\mathbf{a}[i]$ to denote a_i and $\{\mathbf{a}\}$ to denote the set of variables which occur in \mathbf{a} . For every tuple $\mathbf{i} = (i_1, \dots, i_n) \in [k]^n$ we use $\pi_{\mathbf{i}} \mathbf{a}$ to denote the *projection* of \mathbf{a} to \mathbf{i} , i.e, the tuple $(a_{i_1}, \dots, a_{i_n})$. If $I \subseteq [k]$ we might abuse slightly notation and use $\pi_I \mathbf{a}$ to refer to $\pi_{\mathbf{i}} \mathbf{a}$ where \mathbf{i} is the tuple that contains the elements of I in increasing order. For every function f on a domain containing $\{\mathbf{a}\}$, we denote by $f(\mathbf{a})$ the coordinate-wise application of f to \mathbf{a} .

Given a set A and a positive integer k , a k -ary *relation* over A is a subset of A^k . A *signature* σ is a finite collection of relation symbols, each with an associated arity. We shall use $\text{arity}(R)$ to denote the arity of a relation symbol R . A *relational structure* \mathbf{A} over σ , or simply a σ -structure, consists of a set A called the *universe* of \mathbf{A} , and a relation $R(\mathbf{A})$ over A for each $R \in \sigma$ of the corresponding arity. We denote by $\mathcal{C}_{\mathbf{A}}$ the set $\{(\mathbf{a}, R) \mid \mathbf{a} \in R(\mathbf{A}), R \in \sigma\}$. Elements (\mathbf{a}, R) of $\mathcal{C}_{\mathbf{A}}$ will alternatively be denoted $R(\mathbf{a})$ and will be referred to as *constraints*. We shall usually use the same boldface and (standard) capital letter to refer to a structure and its universe, respectively.

A *graph* is a relational structure whose signature consists of a single binary relation that is symmetric and non-reflexive.

Let \mathbf{A}, \mathbf{B} be σ -structures. A *homomorphism* from \mathbf{A} to \mathbf{B} is a map $h : A \rightarrow B$ such that for every $R \in \sigma$ and every $\mathbf{a} \in R(\mathbf{A})$ it holds that $h(\mathbf{a}) \in R(\mathbf{B})$. If there exists a homomorphism from \mathbf{A} to \mathbf{B} we say that \mathbf{A} is homomorphic to \mathbf{B} and we write $\mathbf{A} \rightarrow \mathbf{B}$. We shall use $\text{Hom}(\mathbf{A}; \mathbf{B})$ to denote the number of homomorphisms from \mathbf{A} to \mathbf{B} . The problem of deciding, given two similar structures \mathbf{A} and \mathbf{B} , whether \mathbf{A} is homomorphic to \mathbf{B} is known as the *Constraint Satisfaction Problem* (CSP). If we fix the target structure \mathbf{B} so that the input is only \mathbf{A} , then we obtain the Constraint Satisfaction Problem over \mathbf{B} , denoted $\text{CSP}(\mathbf{B})$.

An *isomorphism* from \mathbf{A} to \mathbf{B} is a bijective map $f : A \rightarrow B$ such that for every $R \in \sigma$ and every $\mathbf{a} \in A^{\text{arity}(R)}$ it holds that $\mathbf{a} \in R(\mathbf{A})$ if and only if $f(\mathbf{a}) \in R(\mathbf{B})$.

The *union* $\mathbf{A} \cup \mathbf{B}$ of two σ -structures \mathbf{A} and \mathbf{B} is the structure \mathbf{C} with $C = A \cup B$ and $R(\mathbf{C}) = R(\mathbf{A}) \cup R(\mathbf{B})$ for every $R \in \sigma$. The *disjoint union* of two structures \mathbf{A} and \mathbf{B} is the structure $\mathbf{A} \cup \mathbf{C}$ where \mathbf{C} is any σ -structure isomorphic to \mathbf{B} satisfying $A \cap C = \emptyset$. We say that a structure is *connected* if it cannot be expressed as the disjoint union of two structures. We say that \mathbf{A} is a *substructure* of \mathbf{B} if $\mathbf{A} \cup \mathbf{B} = \mathbf{B}$. If, in addition, $R(\mathbf{A}) = R(\mathbf{B}) \cap A^{\text{arity}(R)}$ for every $R \in \sigma$ then \mathbf{A} is the substructure of \mathbf{B} *induced* by A .

Tree-like structures

We define the *factor graph*¹ of a structure \mathbf{A} to be the bipartite graph with nodes $A \cup \mathcal{C}_{\mathbf{A}}$ and where every $R(\mathbf{a})$ in $\mathcal{C}_{\mathbf{A}}$ is joined by an edge with every $a \in \{\mathbf{a}\}$.

Then, we say that a structure \mathbf{T} is an *ftree* (factor tree) if its factor graph is a tree in the ordinary graph-theoretic sense. If \mathbf{Q} is a substructure of \mathbf{T} and \mathbf{Q} is an ftree then we say that \mathbf{Q} is a *subftree* of \mathbf{T} .

A *tree-decomposition* of a structure \mathbf{A} is a pair (G, β) where $G = (V, E)$ is a tree and $\beta : V \rightarrow \mathcal{P}(A)$ is a mapping such that the following conditions are satisfied:

1. For every constraint $R(\mathbf{a})$ in $\mathcal{C}_{\mathbf{A}}$ there exists a node $v \in V$ such that $\{\mathbf{a}\} \subseteq \beta(v)$
2. If $a \in \beta(u) \cap \beta(v)$ then $a \in \beta(w)$ for every node w in the unique path in G joining u to v .

The *width* of a tree-decomposition (G, β) is $\max\{|\beta(v)| \mid v \in V\} - 1$ and the *treewidth* of \mathbf{A} is defined as the smallest width among all its tree-decompositions.

The Sherali–Adams hierarchy

In this presentation we follow [3]. Let $P \subseteq [0, 1]^n$ be a polytope $\{\mathbf{x} \in \mathbb{R}^n : M\mathbf{x} \geq \mathbf{b}, 0 \leq \mathbf{x} \leq 1\}$ for a matrix $M \in \mathbb{R}^{m \times n}$, and a column vector $\mathbf{b} \in \mathbb{R}^m$. We denote the convex hull of the $\{0, 1\}$ -vectors in P by $P^{\mathbb{Z}}$. The sequence of Sherali-Adams relaxations of $P^{\mathbb{Z}}$ is the sequence of polytopes $P = P^1 \supseteq P^2 \supseteq \dots$ where P^k is defined in the following way.

Each inequality in $M\mathbf{x} \geq \mathbf{b}$ is multiplied by all possible terms of the form $\prod_{i \in I} x_i \prod_{j \in J} (1 - x_j)$ where $I, J \subseteq [n]$ satisfy $|I \cup J| \leq k - 1$ and $I \cap J = \emptyset$. This leaves a system of polynomial inequalities, each of degree at most k . Then, this system is *linearized* and hence relaxed in the following way: each square x_i^2 is replaced by x_i and each resulting monomial $\prod_{i \in K} x_i$ is replaced by a variable y_K . In this way we obtain a polytope P_L^k . Finally, P_L^k is projected back to n dimensions by defining

$$P^k := \{\mathbf{x} \in \mathbb{R}^n : \text{there exists } \mathbf{y} \in P_L^k \text{ such that } \mathbf{y}_{\{i\}} = \mathbf{x}_i \text{ for each } i \in [n]\}.$$

We note here that $P^{\mathbb{Z}} \subseteq P^k$ for every $k \geq 1$.

¹ We note that the definition of factor graph presented here differs from that in [9], in which the edges are labelled. We also note that the notion of factor graph, although similar, differs in several ways from the incidence multigraph (see [25]) as the latter allows for parallel edges.

In order to apply the SA method to the homomorphism problem there are different possible choices for the polytope P (encoding a relaxation of homomorphism) to start with, each one then yielding a different hierarchy.

Here we shall adapt a SA-based family of relaxations commonly used in optimization variants of CSP [11, 17, 33, 36] which we transform into a relaxation of (plain) CSP by just turning the objective function into a set of new restrictions. Hence, the resulting system of inequalities is not, strictly speaking, obtained using the SA method. Nonetheless, we shall abuse slightly notation and still use SA^k to refer to our system of inequalities.

In fact, giving an explicit description of all inequalities obtained using the SA method for any natural polytope P encoding the LP relaxation for a *general* CSP in our setting is a bit cumbersome (because the constraints of the CSP are encoded in the polytope-defining inequalities instead of the objective function as in CSP optimization variants). Hence, it seems sensible to settle for a good approximation as SA^k . Indeed, as it can be seen in Appendix A, the sequence of relaxations SA^k is tightly interleaved with the sequence P^k obtained by the SA method, in stricto sensu, for a natural choice of initial polytope P .

Given two structures \mathbf{A} and \mathbf{B} , the system of inequality $\text{SA}^k(\mathbf{A}, \mathbf{B})$ for the homomorphism problem over (\mathbf{A}, \mathbf{B}) contains a variable $p_V(f)$ for every $V \subseteq A$ with $1 \leq |V| \leq k$ and every $f : V \rightarrow B$, and a variable $p_{R(\mathbf{a})}(f)$ for every $R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$ and every $f : \{\mathbf{a}\} \rightarrow B$. Each variable must take a value in the range $[0, 1]$. The variables are constrained by the following conditions:

$$\sum_{f:V \rightarrow B} p_V(f) = 1 \quad V \subseteq A \text{ s.t. } |V| \leq k \quad (\text{SA1})$$

$$p_U(f) = \sum_{g:V \rightarrow B, g|_U=f} p_V(g) \quad U \subseteq V \subseteq A \text{ s.t. } |V| \leq k, f : U \rightarrow B \quad (\text{SA2})$$

$$p_U(f) = \sum_{g:V \rightarrow B, g|_U=f} p_{R(\mathbf{a})}(g) \quad R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}, U \subseteq \{\mathbf{a}\} = V \text{ s.t. } |U| \leq k, f : U \rightarrow B \quad (\text{SA3})$$

$$p_{R(\mathbf{a})}(f) = 0 \quad R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}, f : \{\mathbf{a}\} \rightarrow B \text{ s.t. } f(\mathbf{a}) \notin R(\mathbf{B}) \quad (\text{SA4})$$

For the particular case of $k = 1$ we shall use the simplified notation $p_v(f(v))$ to denote the variable $p_V(f)$ for a singleton set $V = \{v\}$ and a function $f : V \rightarrow B$.

The transformation $*_k$

For every $k > 0$ we define an operator $*_k$ that maps a structure into a new structure. As we shall see later this operator will allow to reduce SA^k feasibility to SA^1 feasibility.

Let \mathbf{A} be a σ -structure. Then we define the universe of \mathbf{A}_k^* to be $A_k^* := \cup_{j \leq k} A^j \cup \mathcal{C}_{\mathbf{A}}$. Additionally, \mathbf{A}_k^* contains the following relations:

$$\begin{aligned} T_{j,S}(\mathbf{A}_k^*) &= \{(a_1, \dots, a_j) \in A^j \mid a_i = a_{i'} \forall i, i' \in S\} & j \leq k, S \subseteq [j] \\ T_{j,\mathbf{i}}(\mathbf{A}_k^*) &= \{(\mathbf{a}, \pi_{\mathbf{i}} \mathbf{a}) \mid \mathbf{a} \in A^j\} & j', j \leq k, \mathbf{i} \in [j]^{j'} \\ R_S(\mathbf{A}_k^*) &= \{(a_1, \dots, a_{\text{arity}(R)}) \in R(\mathbf{A}) \mid a_i = a_{i'} \forall i, i' \in S\} & R \in \sigma, S \subseteq [\text{arity}(R)] \\ R_{\mathbf{i}}(\mathbf{A}_k^*) &= \{(R(\mathbf{a}), \pi_{\mathbf{i}} \mathbf{a}) \mid \mathbf{a} \in R(\mathbf{A})\} & R \in \sigma, j \leq k, \mathbf{i} \in [\text{arity}(R)]^j. \end{aligned}$$

Then we have:

► **Lemma 1.** *Let \mathbf{A}, \mathbf{B} be σ -structures. Then $\text{SA}^k(\mathbf{A}, \mathbf{B})$ is feasible if and only if $\text{SA}^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$ is feasible.*

Iterated degree, fractional isomorphism, and equitable partitions

In order to prove our main result we need to lift the known equivalence between fractional isomorphism and the WL algorithm from graphs to relational structures.

Let $L := \{(S, R) \mid R \in \sigma, S \subseteq [\text{arity}(R)]\}$ be a set, the elements of which we shall call *labels*. We construct the *matrix representation* $M_{\mathbf{A}}$ of a σ -structure \mathbf{A} as follows (it will be convenient to assume that the indices of the rows and columns of a matrix are arbitrary sets). $M_{\mathbf{A}}$ is an $A \times \mathcal{C}_{\mathbf{A}}$ matrix whose entries are elements of L . In particular, for all $a \in A$ and $R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$, we have that $M_{\mathbf{A}}[a, R(\mathbf{a})] = (S, R)$ where S is the set containing all elements $i \in [\text{arity}(R)]$ such that $a = \mathbf{a}[i]$.

In a nutshell we are lifting from graph isomorphism to matrix isomorphism, where two matrices are isomorphic if they are identical modulo a permutation of the rows and columns. To formalize this, it will be convenient to associate \mathbf{A} with a set of 0-1 incidence matrices. In particular, for every $\ell \in L$ we define $M_{\mathbf{A}}^{\ell} \in \{0, 1\}^{A \times \mathcal{C}_{\mathbf{A}}}$ as follows: $M_{\mathbf{A}}^{\ell}[a, R(\mathbf{a})] = 1$ if $M_{\mathbf{A}}[a, R(\mathbf{a})] = \ell$ and $M_{\mathbf{A}}^{\ell}[a, R(\mathbf{a})] = 0$ otherwise. In [19], relaxations of matrix isomorphism are also considered although in that setting the matrices have real entries and the goal is different from ours.

We now describe a procedure akin to the 1-dimensional Weisfeiler-Leman algorithm to calculate iterative refinements of a colouring of the universe and constraint set of a relational structure. While there are syntactical differences, when run on graphs this procedure is equivalent for all purposes to 1-WL. For every $k \geq 0$ and $x \in A \cup \mathcal{C}_{\mathbf{A}}$, we define inductively the *iterated degree* $\delta_k^{\mathbf{A}}(x)$ of x on \mathbf{A} as follows. We set $\delta_0^{\mathbf{A}}(x)$ to be one of two arbitrary symbols that distinguish elements of A from elements of $\mathcal{C}_{\mathbf{A}}$. For $k \geq 1$ we set $\delta_k^{\mathbf{A}}(a) = \{ \{ (\ell, \delta_{k-1}^{\mathbf{A}}(\mathbf{a}, R)) \mid M_{\mathbf{A}}^{\ell}[a, R(\mathbf{a})] = 1 \} \}$ and $\delta_k^{\mathbf{A}}(\mathbf{a}, R) = \{ \{ (\ell, \delta_{k-1}^{\mathbf{A}}(a)) \mid M_{\mathbf{A}}^{\ell}[a, R(\mathbf{a})] = 1 \} \}$, where double curly brackets denote that $\delta_k^{\mathbf{A}}(x)$ is a multiset.

We say that \mathbf{A} and \mathbf{B} have the same iterated degree sequence if for every $k \geq 0$, $\{ \{ \delta_k^{\mathbf{A}}(a) \mid a \in A \cup \mathcal{C}_{\mathbf{A}} \} \} = \{ \{ \delta_k^{\mathbf{B}}(b) \mid b \in B \cup \mathcal{C}_{\mathbf{B}} \} \}$. Note that if there exists a matrix isomorphism from \mathbf{A} to \mathbf{B} (or alternatively, \mathbf{A} and \mathbf{B} are isomorphic) then \mathbf{A} and \mathbf{B} have the same iterated degree sequence, but the converse does not hold.

The notion of equitable partition is key in the proof of the equivalence of the different characterizations of fractional isomorphism. We present its adaptation to relational structures. A *partition* of a σ -structure \mathbf{A} is a pair (P, Q) where $P = \{P_i \mid i \in I\}$ is a partition of A and $Q = \{Q_j \mid j \in J\}$ is a partition of $\mathcal{C}_{\mathbf{A}}$. We say that (P, Q) is *equitable* if for every $i \in I$, $j \in J$, and $\ell \in L$, there are integers $c_{i,j}^{\ell}, d_{j,i}^{\ell}$, called the *parameters* of the partition, such that for every every $i \in I$, every $a \in P_i$, every $\ell \in L$, and every $j \in J$, we have

$$|\{(\mathbf{a}, R) \in Q_j \mid M_{\mathbf{A}}[a, R(\mathbf{a})] = \ell\}| = c_{i,j}^{\ell} \quad (\text{P1})$$

and, similarly, for every $j \in J$, every $R(\mathbf{a}) \in Q_j$, every $\ell \in L$, and every $i \in I$ we have

$$|\{a \in P_i \mid M_{\mathbf{A}}[a, (\mathbf{a}, R)] = \ell\}| = d_{j,i}^{\ell}. \quad (\text{P2})$$

We say that two structures \mathbf{A}, \mathbf{B} have a *common equitable partition* if there are equitable partitions $(\{P_i^{\mathbf{A}} \mid i \in I\}, \{Q_j^{\mathbf{A}} \mid j \in J\})$ and $(\{P_i^{\mathbf{B}} \mid i \in I\}, \{Q_j^{\mathbf{B}} \mid j \in J\})$ of \mathbf{A} and \mathbf{B} with the same parameters satisfying $|P_i^{\mathbf{A}}| = |P_i^{\mathbf{B}}|$ for every $i \in I$ and $|Q_j^{\mathbf{A}}| = |Q_j^{\mathbf{B}}|$ for every $j \in J$. We note that if \mathbf{A} and \mathbf{B} are connected it is not necessary to verify this latter requirement and, instead, it is enough to check that $|A| = |B|$.

A matrix M of non-negative real numbers is said to be *left* (resp. *right*) *stochastic* if all its columns (resp. rows) sum to 1. Note that we do not require M to be square. A *doubly stochastic matrix* is a square matrix that is both left and right stochastic.

► **Theorem 2.** *Let \mathbf{A}, \mathbf{B} be σ -structures. The following are equivalent:*

1. *There exist doubly stochastic matrices X, Y such that $XM_{\mathbf{A}}^{\ell} = M_{\mathbf{B}}^{\ell}Y$ and $M_{\mathbf{A}}^{\ell}Y^T = X^T M_{\mathbf{B}}^{\ell}$ for every $\ell \in L$;*
2. *\mathbf{A} and \mathbf{B} have the same iterated degree sequence;*
3. *\mathbf{A} and \mathbf{B} have a common equitable partition;*
4. *$\text{Hom}(\mathbf{T}, \mathbf{A}) = \text{Hom}(\mathbf{T}, \mathbf{B})$ for all σ -ftrees \mathbf{T} .*

If, additionally, \mathbf{A} and \mathbf{B} are graphs, then the following is also equivalent:

5. *There exists a doubly stochastic matrix X such that $XN_{\mathbf{A}} = N_{\mathbf{B}}X$ where $N_{\mathbf{A}}$ and $N_{\mathbf{B}}$ denote the adjacency matrices of \mathbf{A} and \mathbf{B} respectively.*

We refer the reader to the full version of this paper for the proof. The equivalence between (1), (2) and (3) is an immediate generalization of the equivalence between the algebraic and combinatorial characterizations of fractional graph isomorphism (see for example [31]), while (4) generalizes the corresponding characterization of graphs from [14] in terms of counting homomorphisms from trees. Also, note that condition (5) shows that our definition coincides with the standard notion of fractional isomorphism when \mathbf{A} and \mathbf{B} are graphs.

Similarly to the case of graphs, the notion of fractional isomorphism captured in Theorem 2 can be strengthened giving rise to a hierarchy of increasingly tighter relaxations. In particular, for every $k \geq 1$, we shall denote $\mathbf{A} \equiv_k \mathbf{B}$ whenever \mathbf{A}_k^* and \mathbf{B}_k^* satisfy the conditions of Theorem 2.

It is easy to see that the case $k = 1$ would be unchanged if one replaces \mathbf{A}_1^* and \mathbf{B}_1^* by \mathbf{A} and \mathbf{B} respectively and, hence, Theorem 2 characterizes \equiv_1 . For other small values of k other than $k = 1$, \equiv_k is a bit more difficult to characterize. However, as long as k is at least as large as the arity of any relation in the signature then we have the following result.

► **Lemma 3.** *Let r be the maximum arity among all relations in σ and assume that $r \leq k$. Then for every pair of structures \mathbf{A}, \mathbf{B} the following are equivalent:*

1. $\mathbf{A} \equiv_k \mathbf{B}$
2. $\text{Hom}(\mathbf{Q}; \mathbf{A}) = \text{Hom}(\mathbf{Q}; \mathbf{B})$ for every structure \mathbf{Q} of treewidth $< k$.

This result, which follows easily from condition (4) in Theorem 2, is inspired by a similar result [14] which states that two graphs \mathbf{A}, \mathbf{B} are indistinguishable by the k -WL algorithm if and only if $\text{Hom}(\mathbf{Q}; \mathbf{A}) = \text{Hom}(\mathbf{Q}; \mathbf{B})$ for every graph \mathbf{Q} of treewidth $\leq k$. It is not that surprising that a similar result can be shown for \equiv_k since, after all, the k -WL algorithm can be seen as the 1-WL algorithm applied to k -ary tuples. However, note that the bound on the treewidth differs in one unit between k -WL and \equiv_k . Still, using Lemma 3 it can be shown that for $r \leq k$, \equiv_k can be alternatively characterized in logical terms extending, again, an analogous result for k -WL [22]. More precisely, $\mathbf{A} \equiv_k \mathbf{B}$ if and only if \mathbf{A} and \mathbf{B} satisfy the same formulas in the k -variable fragment of first-order logic with counting quantifiers, denoted C^k (we omit the definition as it will not be needed).

2.1 Main results

The main result of this paper is a new characterization in terms of fractional isomorphism of the Sherali-Adams relaxation of the homomorphism problem.

► **Theorem 4.** *Let \mathbf{A}, \mathbf{B} be relational structures. Then, the following are equivalent:*

1. $\text{SA}^k(\mathbf{A}, \mathbf{B})$ is feasible;
2. *There exists a sequence of structures $\mathbf{X}_0, \dots, \mathbf{X}_n$ such that $\mathbf{X}_0 = \mathbf{A}_k^*$, $\mathbf{X}_n = \mathbf{B}_k^*$, and for all $i = 0, \dots, n-1$ we have that $\mathbf{X}_i \rightarrow \mathbf{X}_{i+1}$ or $\mathbf{X}_i \equiv_1 \mathbf{X}_{i+1}$;*
3. *There exist structures $\mathbf{X}_1, \mathbf{X}_2$ such that $\mathbf{A}_k^* \rightarrow \mathbf{X}_1$, $\mathbf{X}_1 \equiv_1 \mathbf{X}_2$, and $\mathbf{X}_2 \rightarrow \mathbf{B}_k^*$.*

Theorem 4 is an immediate consequence, using Lemma 1, of the following Theorem.

► **Theorem 5.** *Let \mathbf{A}, \mathbf{B} be relational structures. Then, the following are equivalent:*

1. $\text{SA}^1(\mathbf{A}, \mathbf{B})$ is feasible;
2. There exist left stochastic matrices X, Y such that for every $\ell = (S, R) \in L$, it holds that $XM_{\mathbf{A}}^{\ell} \leq \sum_{\ell'} M_{\mathbf{B}}^{\ell'} Y$ where ℓ' ranges over all $(S', R) \in L$ with $S \subseteq S'$;
3. There exists a sequence of structures $\mathbf{X}_0, \dots, \mathbf{X}_n$ such that $\mathbf{X}_0 = \mathbf{A}$, $\mathbf{X}_n = \mathbf{B}$, and for all $i = 0, \dots, n-1$ we have that $\mathbf{X}_i \rightarrow \mathbf{X}_{i+1}$ or $\mathbf{X}_i \equiv_1 \mathbf{X}_{i+1}$;
4. There exist structures $\mathbf{X}_1, \mathbf{X}_2$ such that $\mathbf{A} \rightarrow \mathbf{X}_1$, $\mathbf{X}_1 \equiv_1 \mathbf{X}_2$, and $\mathbf{X}_2 \rightarrow \mathbf{B}$.

If in addition \mathbf{A} and \mathbf{B} have no loops (meaning that there are no repeated elements in any constraint) then the following condition is also equivalent:

5. There exist left stochastic matrices X, Y such that for every $\ell \in L$ it holds that $XM_{\mathbf{A}}^{\ell} = M_{\mathbf{B}}^{\ell} Y$.

Note that for graphs condition (5) of the above theorem is naturally seen as the homomorphism counterpart of the notion of fractional isomorphism (see condition (1) in Lemma 2). Consequently, we shall say that \mathbf{A} is fractionally homomorphic to \mathbf{B} whenever \mathbf{A} and \mathbf{B} satisfy the conditions of Theorem 5.

While Theorems 4 and 5 have applications, for instance in the field of Constraint Satisfaction Problems (see Section 4), we believe that their main interest is that they shed light on the LP relaxations for homomorphism as they show that fractional homomorphism and its higher order counterparts can be decomposed into a sequence of basic, better studied morphisms, a fact which we find interesting on its own. Additionally, they establish a close link between LP relaxations for isomorphism and homomorphism, which were introduced initially in different fields.

3 Proof of Theorem 5

The equivalence (1) \Leftrightarrow (2) is merely syntactic. In particular we shall show that there is a one-to-one satisfiability-preserving correspondence between pairs of matrices and variable assignments of $\text{SA}^1(\mathbf{A}, \mathbf{B})$. However, we first need to massage a bit the two formulations. First, we can assume that for every $R^A(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$ and $R^B(\mathbf{b}) \in \mathcal{C}_{\mathbf{B}}$, the corresponding entry in Y is null unless $R^A = R^B$ and $f(\mathbf{a}) = \mathbf{b}$ for some $f : \{\mathbf{a}\} \rightarrow \{\mathbf{b}\}$, since otherwise there is no way that Y can be part of a feasible solution. Secondly, we note that the feasibility of $\text{SA}^1(\mathbf{A}, \mathbf{B})$ does not change if in (SA3) we replace $=$ by \leq obtaining a new set of inequalities (which to avoid confusion we shall denote by (SA3')) and, in addition, we add for every $R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$ the equality

$$\sum_{f: \{\mathbf{a}\} \rightarrow B} p_{R(\mathbf{a})}(f) = 1. \quad (\text{SA5})$$

Finally, note that in $\text{SA}^1(\mathbf{A}, \mathbf{B})$ we can ignore (SA2).

Then we can establish the following correspondence between pairs of matrices X, Y and assignments $\text{SA}^1(\mathbf{A}, \mathbf{B})$: for every $a \in A$ and $b \in B$, we set $p_a(b) = X[b, a]$ and for every $R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$ and $f : \{\mathbf{a}\} \rightarrow B$ we define $p_{R(\mathbf{a})}(f) = Y[R(f(\mathbf{a})), R(\mathbf{a})]$. Then, it is easy to see that (SA3') corresponds to $XM_{\mathbf{A}}^{\ell} \leq \sum_{\ell' \in L_{\ell}} M_{\mathbf{B}}^{\ell'} Y$ for every $\ell \in L$ (where $L_{(S,R)} := \{(S', R) \in L \mid S \subseteq S'\}$), X being left stochastic corresponds to (SA1), and Y being left stochastic corresponds to (SA5).

The equivalence (1) \Leftrightarrow (5) is obtained as in (1) \Leftrightarrow (2). We just need to notice that when \mathbf{A} and \mathbf{B} have no loops, then none of the entries in $M_{\mathbf{A}}$ and $M_{\mathbf{B}}$ contain any label $\ell = (S, R) \in L$ where $|S| > 1$ and hence it is only necessary to consider labels $\ell = (S, R) \in L$ where S is a singleton. Observe that, in this case, the equation in (2) becomes $XM_{\mathbf{A}}^{\ell} \leq M_{\mathbf{B}}^{\ell} Y$

since for every label $\ell = (S, R)$ where S is a singleton, the only label (S', R) with $S \subseteq S'$ and $M_{\mathbf{B}}^{\ell}$ not a zero matrix is ℓ itself. Finally, in order to replace \leq by $=$ in the previous equation we just need to use (SA3) instead of (SA3').

Notice that (4) \implies (3) is trivial.

The proof of (3) \implies (2) is by induction on n . If $n = 0$ the claim is immediate, so assume that $n \geq 1$. Let $\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_n$ be a sequence of structures satisfying (3). By the induction hypothesis, there exist left stochastic matrices X, Y such that $XM_{\mathbf{X}_1}^{\ell} \leq \sum_{\ell' \in L_{\ell}} M_{\mathbf{X}_n}^{\ell'} Y$ for all $\ell \in L$.

If $\mathbf{X}_0 \equiv_1 \mathbf{X}_1$ then it follows from Theorem 2 that there exist doubly stochastic matrices X' and Y' such that $X'M_{\mathbf{X}_0}^{\ell} = M_{\mathbf{X}_1}^{\ell} Y'$ for all $\ell \in L$, and so it is easy to verify that XX', YY' are such that (2) holds. Assume that $\mathbf{X}_0 \not\equiv_1 \mathbf{X}_1$. We shall show that there exist left stochastic matrices X' and Y' such that for all $b \in B$, for all $R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$, and for all $\ell \in L$ there exists $\hat{\ell} = \hat{\ell}(b, R(\mathbf{a}), \ell) \in L_{\ell}$ such that $XA^{\ell}[b, R(\mathbf{a})] \leq B^{\hat{\ell}}Y[b, R(\mathbf{a})]$. Assuming that this holds, again it follows by the induction hypothesis that XX', YY' are left stochastic matrices such that $XX'M_{\mathbf{X}_0}^{\ell} \leq \sum_{\ell' \in L_{\ell}} M_{\mathbf{X}_n}^{\ell'} YY'$ for all $\ell \in L$.

Let h be a homomorphism from \mathbf{A} to \mathbf{B} . We define $X'[b, a] = 1$ if $b = h(a)$ and $X'[b, a] = 0$ otherwise. Similarly, we set $Y'[R^B(\mathbf{b}), R^A(\mathbf{a})] = 1$ if $\mathbf{b} = h(\mathbf{a})$ and $R^B = R^A$ and $Y'[R^B(\mathbf{b}), R^A(\mathbf{a})] = 0$ otherwise. It is easy to see that X' and Y' are left stochastic. Now let $\ell = (S, R) \in L$, $b \in B$ and $R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$. If $M_{\mathbf{A}}^{\ell}[a, R(\mathbf{a})] = 0$ for all $a \in A$ then $XM_{\mathbf{A}}^{\ell}[b, R(\mathbf{a})] = 0$ and there is nothing to prove. So we can assume that there is $a \in A$ such that for all $i \in [\text{arity}(R)]$, $\mathbf{a}[i] = a$ if and only if $i \in S$. Then we have that $XM_{\mathbf{A}}^{\ell}[b, R(\mathbf{a})] = 1$ if $b = h(a)$, and $XM_{\mathbf{A}}^{\ell}[b, R(\mathbf{a})] = 0$ otherwise. Again in the latter case there is nothing to prove so let us assume that $b = h(a)$. It follows that $h(\mathbf{a}[i]) = b$ for all $i \in S$ and hence there exists $\hat{\ell} = (R, S')$ with $S \subseteq S'$ such that $M_{\mathbf{B}}^{\hat{\ell}}[b, R(h(\mathbf{a}))] = 1$, which completes the proof.

It only remains to prove (1) \implies (4). Assume that \mathbf{A} and \mathbf{B} satisfy (1). Further, we can assume that there exists an integer $m > 0$ such that all variables in the feasible solution of $\text{SA}^1(\mathbf{A}, \mathbf{B})$ take rational values of the form n/m for some integer n . Let Y be the set of all tuples $((b_1, c_1), \dots, (b_m, c_m)) \in (B \times [m])^m$ satisfying the following conditions:

- $(b_i, c_i) \neq (b_{i'}, c_{i'})$ for every $i \neq i' \in [m]$ (i.e., the tuple has no repeated elements), and
- for every $i \in [m]$, c_i is at most $|\{j \in [m] \mid b_j = b_i\}|$.

In other words, Y is the set of tuples that can be obtained if in every tuple $(b_1, \dots, b_m) \in B^m$ we replace all occurrences, say n , of any symbol $b \in B$ by n ‘‘copies’’ $(b, 1), \dots, (b, n)$, in any possible order. Let $\Pi : Y \rightarrow B^m$ be the function mapping $((b_1, c_1), \dots, (b_m, c_m))$ to (b_1, \dots, b_m) .

Let $X = A \times Y$. For every permutation τ on $[m]$ and any m -ary tuple $\mathbf{z} = (z_1, \dots, z_m) \in Z^m$ where Z is any arbitrary set we shall use $\mathbf{z} \circ \tau$ to denote the tuple $(z_{\tau(1)}, \dots, z_{\tau(m)})$. This notation is justified by the fact that we can see formally \mathbf{z} as a mapping from $[m]$ to Z . For any $x = (a, \mathbf{y}) \in X$, we shall abuse notation and write $x \circ \tau$ to denote $(a, \mathbf{y} \circ \tau)$.

For any two \mathbf{z}, \mathbf{z}' in any of the sets B^m, Y or X , we shall write $\mathbf{z} \sim \mathbf{z}'$ iff there exists some permutation τ on $[m]$ such that $\mathbf{z}' = \mathbf{z} \circ \tau$.

For every multiset $U = \{\{b_1, \dots, b_m\}\}$ of m elements from B , we shall fix one arbitrary element of Y , denoted $\langle U \rangle$, satisfying that the multiset of variables in $\Pi(\langle U \rangle)$ coincides with U . We shall abuse slightly notation and for every $\mathbf{b} = (b_1, \dots, b_m) \in B^m$ we shall also use $\langle \mathbf{b} \rangle$ to denote $\langle \{\{b_1, \dots, b_m\}\} \rangle$. Clearly if $\mathbf{b} \sim \mathbf{b}'$ then $\langle \mathbf{b} \rangle = \langle \mathbf{b}' \rangle$. Moreover, for every $\mathbf{b} \in B^m$ we shall denote by $\mathbf{y}^{\mathbf{b}}$ the tuple $((b_1, c_1), \dots, (b_m, c_m)) \in Y$ obtained by choosing $c_i < c_j$ whenever $b_i = b_j$ and $i < j$. Notice that $\Pi(\mathbf{y}) = \mathbf{b}$ and $\mathbf{y}^{\mathbf{b}} \sim \langle \mathbf{b} \rangle$.

Now we shall show how to construct two structures \mathbf{X}_1 and \mathbf{X}_2 which satisfy condition (4). The domain of both \mathbf{X}_1 and \mathbf{X}_2 is X . The constraints of \mathbf{X}_1 and \mathbf{X}_2 are constructed as follows. Let J be the set of all triplets (R, T, \mathbf{a}) where $R \in \sigma$, $\mathbf{a} \in A^{\text{arity}(R)}$, and T is a multiset of size m of tuples in $R(\mathbf{B})$ such that for every $s, s' \in [\text{arity}(R)]$:

$$\mathbf{a}[s] = \mathbf{a}[s'] \Rightarrow \forall \mathbf{t} \in T(\mathbf{t}[s] = \mathbf{t}[s']).$$

In other words, T only contains tuples $\mathbf{t} \in R(\mathbf{B})$ such that $\mathbf{t} = f(\mathbf{a})$ for some function $f : \{\mathbf{a}\} \rightarrow B$. Then for every $(R, T, \mathbf{a}) \in J$ and $d \in [2]$, we define Q_j^d to be the set of $m!$ constraints obtained in the following way. Fix any arbitrary ordering $\mathbf{t}_1, \dots, \mathbf{t}_m$ of the tuples in T and let $\mathbf{b}_1, \dots, \mathbf{b}_r \in B^m$, $r = \text{arity}(R)$ such that $\mathbf{b}_s[i] = \mathbf{t}_i[s]$ for every $i \in [m]$ and $s \in [r]$. Then, for every permutation $\tau \in [m]$, we include constraint $R((a_1, \langle \mathbf{b}_1 \rangle) \circ \tau, \dots, (a_r, \langle \mathbf{b}_r \rangle) \circ \tau)$ in Q_j^1 and constraint $R((a_1, \mathbf{y}^{\mathbf{b}_1}) \circ \tau, \dots, (a_r, \mathbf{y}^{\mathbf{b}_r}) \circ \tau)$ in Q_j^2 , where $\mathbf{a} = (a_1, \dots, a_r)$.

Finally, we define $C_{\mathbf{X}_d}$, $d \in [2]$ to be $\cup_{j \in J} Q_j^d$ (note that for each $d \in [2]$, all sets Q_j^d , $j \in J$ are disjoint). To complete the proof it only remains to show that \mathbf{X}_1 and \mathbf{X}_2 satisfy condition (4).

▷ **Claim 6.** \mathbf{A} is homomorphic to \mathbf{X}_1 .

Proof. Let p be a feasible solution of $\text{SA}^1(\mathbf{A}, \mathbf{B})$. We define a mapping $h : A \rightarrow X$ by setting $h(a) = (a, \langle \mathbf{c}_a \rangle)$, where $\mathbf{c}_a \in B^m$ is any tuple satisfying that every element $b \in B$ occurs exactly $m \cdot p_a(b)$ times. We shall show that h is a homomorphism from \mathbf{A} to \mathbf{X}_1 . Let $R \in \sigma$, $\mathbf{a} = (a_1, \dots, a_r) \in R(\mathbf{A})$ where $r = \text{arity}(R)$, and consider the multiset $T = \{\mathbf{t}_1, \dots, \mathbf{t}_m\}$ of tuples in $R(\mathbf{B})$ obtained by picking each tuple $\mathbf{t}_i = f_i(\mathbf{a}) \in R(\mathbf{B})$ exactly $m \cdot p_{R(\mathbf{a})}(f_i)$ times (note that we are using implicitly (SA4) to guarantee the existence of T). For every $s, s' \in [r]$ satisfying $a_s = a_{s'}$ it follows from the construction of T that $\mathbf{t}[s] = \mathbf{t}[s']$ for every $\mathbf{t} \in T$, and hence $(R, T, \mathbf{a}) \in J$. Let $\mathbf{t}_1, \dots, \mathbf{t}_m$ be the ordering of the elements in T associated to (R, T, \mathbf{a}) in the construction of \mathbf{X}_1 and $\mathbf{b}_1, \dots, \mathbf{b}_r$ be the tuples obtained by setting $\mathbf{b}_s[i] = \mathbf{t}_i[s]$ for $i \in [m]$ and $s \in [r]$. Then, it follows that \mathbf{X}_1 contains the constraint $R((a_1, \langle \mathbf{b}_1 \rangle), \dots, (a_r, \langle \mathbf{b}_r \rangle))$. It follows again from (SA3) that for every $s \in [r]$ $\mathbf{c}_{a_s} \sim \mathbf{b}_s$ and hence $(a_s, \langle \mathbf{b}_s \rangle)$ is precisely $h(a_s)$ as desired. ◁

▷ **Claim 7.** \mathbf{X}_2 is homomorphic to \mathbf{B} .

Proof. Let $h : X \rightarrow B$ be the function mapping any (a, \mathbf{y}) to $\pi_1(\Pi(\mathbf{y}))$. It is immediate that h defines a homomorphism from \mathbf{X}_2 to \mathbf{B} . ◁

▷ **Claim 8.** \mathbf{X}_1 and \mathbf{X}_2 have a common equitable partition.

Proof. We shall prove that (P, Q^1) and (P, Q^2) define a common equitable partition of \mathbf{X}_1 and \mathbf{X}_2 where $P = \{P_i \mid i \in I\}$ is the partition given by the equivalence relation \sim on X and $Q^d = \{Q_j^d \mid j \in J\}$, $d \in [2]$ are as defined in the construction of \mathbf{X}_1 and \mathbf{X}_2 .

This follows immediately from the following fact. Let $d \in [2]$, $i \in I$, $j \in J$, and $\ell \in L$. Then exactly one of the following conditions holds:

1. There is no $x \in P_i$ and $C \in Q_j^d$ such that $M_{\mathbf{X}_d}[x, C] = \ell$;
2. There exists a one-to-one correspondence between the elements of P_i and Q_j^d such that for every pair (x, C) of associated elements, $M_{\mathbf{X}_d}[x, C] = \ell$.

Furthermore, for every $i \in I$, $j \in J$, and $\ell \in L$, condition (1) holds for $d = 1$ if and only if it holds for $d = 2$.

Let us prove it. Let $j = (R, T, \mathbf{a})$ and let $\mathbf{b}_1, \dots, \mathbf{b}_r$ and $\mathbf{y}^{\mathbf{b}_1}, \dots, \mathbf{y}^{\mathbf{b}_r}$ be as in the construction of Q_j^d (where $r = \text{arity}(R)$). We first observe that for every $x \in P_i$, every $C = R(x_1, \dots, x_r) \in Q_j^d$, and every permutation τ on $[m]$, $M_{\mathbf{X}_d}[x, C] = \ell \Leftrightarrow M_{\mathbf{X}_d}[x \circ \tau, C \circ \tau] = \ell$ where we use $C \circ \tau$ to denote $R(x_1 \circ \tau, \dots, x_r \circ \tau)$. It then follows that if (1) fails then (2) must hold.

Now it only remains to see that (1) holds for $d = 1$ if and only if (1) holds for $d = 2$. Clearly, this follows immediately if the relation symbol in ℓ is different from R , so we can assume that $\ell = (R, S)$ for some $S \subseteq [r]$. We then note that if there is some pair (x, C) violating (1), then $M_{\mathbf{X}_d}[x \circ \tau, C \circ \tau] = \ell$ for all permutations τ on $[m]$, and hence C can be chosen to be any constraint in Q_j^d . Hence, if we choose $R((a_1, \langle \mathbf{b}_1 \rangle), \dots, (a_r, \langle \mathbf{b}_r \rangle))$ for $d = 1$ and $R((a_1, \mathbf{y}^{\mathbf{b}_1}), \dots, (a_r, \mathbf{y}^{\mathbf{b}_r}))$ for $d = 2$, in order to complete the proof it is enough to show that for every $s, s' \in [r]$, $(a_s, \langle \mathbf{b}_s \rangle) = (a_{s'}, \langle \mathbf{b}_{s'} \rangle) \Leftrightarrow (a_s, \mathbf{y}^{\mathbf{b}_s}) = (a_{s'}, \mathbf{y}^{\mathbf{b}_{s'}})$.

The direction (\Leftarrow) is immediate. For the direction (\Rightarrow) assume that s, s' satisfy the left-hand side. Since $a_s = a_{s'}$ it follows that $\mathbf{b}_s = \mathbf{b}_{s'}$. Since $\mathbf{y}^{\mathbf{b}_s}$ and $\mathbf{y}^{\mathbf{b}_{s'}}$ are determined in a unique way from \mathbf{b}_s and $\mathbf{b}_{s'}$ respectively, we are done. \triangleleft

4 Some Applications

As an immediate consequence of Theorem 4 we obtain that feasibility of the k^{th} Sherali-Adams relaxation of the homomorphism problem is closed under \equiv_k .

► **Corollary 9.** *Let \mathbf{A} , \mathbf{A}' and \mathbf{B} be σ -structures and suppose that $\mathbf{A} \equiv_k \mathbf{B}$. Then, $\text{SA}^k(\mathbf{A}, \mathbf{B})$ is feasible if and only if $\text{SA}^k(\mathbf{A}', \mathbf{B})$ is feasible.*

We note that if the maximum arity on the signature σ is at most k , then the previous corollary can be alternatively stated in the following way: if \mathbf{A} and \mathbf{A}' are C^k -equivalent then $\text{SA}^k(\mathbf{A}, \mathbf{B})$ is feasible if and only if $\text{SA}^k(\mathbf{A}', \mathbf{B})$ is feasible. Then, one could be tempted to use this observation to transfer results from LP relaxations to logical definability. In particular one could infer that if SA^k decides correctly $\text{CSP}(\mathbf{B})$ then it is definable in the logic $C_{\infty, \omega}^k$ which is the extension of C^k consisting of all formulas made from atomic formulas and equality by means of finitary and infinitary conjunctions, negations, and standard and counting quantifiers. However, this is of limited interest as it follows by combining [2] and [6] that $\text{CSP}(\mathbf{B})$ would also be definable in the much weaker logic $L_{\infty, \omega}^{\max(k, 3)}$ where counting quantifiers are not allowed. Consequently, the previous corollary is most likely to find applications in obtaining lower bounds on the Sherali-Adams rank for concrete instances of CSP.

However, the principal novelty in our result is precisely the opposite direction, which leads to an alternative combinatorial characterization of the Sherali-Adams relaxation. A concrete application is the answer to the following question: for which structures \mathbf{B} is $\text{CSP}(\mathbf{B})$ closed under \equiv_1 -equivalence? This question arises in the context of the distributed CSP [9] where the variables and constraints of an instance are distributed among agents which communicate with each other by sending messages through fixed communication channels. In fact, the connection between the Weisfeiler-Leman algorithm and distributed computation goes back to the influential paper of Angluin on networks of processors [1]. For the distributed CSP, one of the most natural configurations for the communication network is essentially identical to the factor graph [16]. It then follows, under some technical requirements (agent anonymity and synchronicity) that any distributed message passing algorithm will necessarily behave in an identical manner on every two input instances that are \equiv_1 -equivalent and, hence, it follows that $\text{CSP}(\mathbf{B})$ can only be solved by a distributed algorithm if $\text{CSP}(\mathbf{B})$ is closed under \equiv_1 -equivalence.

This question was already answered in [9] where it was shown that $\text{CSP}(\mathbf{B})$ is closed under \equiv_1 if and only if \mathbf{B} has symmetric polymorphisms of all arities, where a k -ary symmetric polymorphism of a σ -structure \mathbf{B} is any homomorphism h from \mathbf{B}^k to \mathbf{B} that is invariant under the permutation of its arguments (the reader can safely ignore the definition of symmetric polymorphism as we will be using it as a black-box). The proof in [9] makes use of a result from [24] stating that $\text{CSP}(\mathbf{B})$ has symmetric polymorphisms of all arities if and only if it is solvable by an LP relaxation known as the basic linear programming relaxation (BLP). Although BLP is slightly different from SA^1 , both coincide over instances (\mathbf{A}, \mathbf{B}) where \mathbf{A} has no loops (i.e., every constraint has no repeated elements). It is then very easy to obtain the following characterization:

► **Lemma 10.** *Let \mathbf{B} be a fixed finite σ -structure. The following are equivalent:*

1. $\text{CSP}(\mathbf{B})$ is closed under \equiv_1 -equivalence;
2. SA^1 decides $\text{CSP}(\mathbf{B})$;
3. BLP decides $\text{CSP}(\mathbf{B})$;
4. \mathbf{B} has symmetric polymorphisms of all arities.

Proof (Sketch). (1) \Leftrightarrow (2) follows from Theorem 5 and (3) \Leftrightarrow (4) from [24]. Hence, it is only necessary to verify (2) \Leftrightarrow (3). We use the following fact which follows from the Sparse Incomparability Lemma [29]: for every instance \mathbf{A} of $\text{CSP}(\mathbf{B})$, there exists a structure \mathbf{A}' with no loops such that $\mathbf{A}' \rightarrow \mathbf{A}$ and $\mathbf{A} \rightarrow \mathbf{B}$ iff $\mathbf{A}' \rightarrow \mathbf{B}$. Now, assume that BLP does not solve $\text{CSP}(\mathbf{B})$. This means that there exists a structure \mathbf{A} not homomorphic to \mathbf{B} and such that $\text{BLP}(\mathbf{A}, \mathbf{B})$ is feasible. Now, let \mathbf{A}' be the structure given by the Sparse Incomparability Lemma. Since $\mathbf{A}' \rightarrow \mathbf{A}$ it follows that $\text{BLP}(\mathbf{A}', \mathbf{B})$ is feasible, and, since \mathbf{A}' has no loops, $\text{SA}^1(\mathbf{A}', \mathbf{B})$ is feasible as well. Since \mathbf{A}' is not homomorphic to \mathbf{B} it follows that SA^1 does not solve $\text{CSP}(\mathbf{B})$. The same argument can be used to show the converse although it is not necessary as it also follows immediately by comparing the inequalities of SA^1 and BLP. ◀

References

- 1 Dana Angluin. Local and global properties in networks of processors (extended abstract). In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, page 82–93, New York, NY, USA, 1980. Association for Computing Machinery. doi:10.1145/800141.804655.
- 2 Albert Atserias, Andrei A. Bulatov, and Anuj Dawar. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.*, 410(18):1666–1683, 2009. doi:10.1016/j.tcs.2008.12.049.
- 3 Albert Atserias and Elitza Maneva. Sherali–adams relaxations and indistinguishability in counting logics. *SIAM Journal on Computing*, 42(1):112–137, 2013. doi:10.1137/120867834.
- 4 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 684–697, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2897518.2897542.
- 5 László Babai, Paul Erdős, and Stanley M. Selkow. Random graph isomorphism. *SIAM J. Comput.*, 9(3):628–635, 1980. doi:10.1137/0209047.
- 6 Libor Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, 26(3):923–943, 2016.
- 7 Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna, and Stanislav Zivný. The power of the combined basic linear programming and affine relaxation for promise constraint satisfaction problems. *SIAM J. Comput.*, 49(6):1232–1248, 2020. doi:10.1137/20M1312745.

- 8 A. A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, October 2017. doi:10.1109/FOCS.2017.37.
- 9 Silvia Butti and Victor Dalmau. The complexity of the distributed constraint satisfaction problem. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPICs*, pages 20:1–20:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.STACS.2021.20.
- 10 Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Comb.*, 12(4):389–410, 1992. doi:10.1007/BF01305232.
- 11 Siu On Chan, James R. Lee, Prasad Raghavendra, and David Steurer. Approximate constraint satisfaction requires large LP relaxations. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 350–359. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.45.
- 12 Víctor Dalmau and Andrei A. Krokhin. Robust satisfiability for csp: Hardness and algorithmic results. *ACM Trans. Comput. Theory*, 5(4):15:1–15:25, 2013. doi:10.1145/2540090.
- 13 Víctor Dalmau, Andrei A. Krokhin, and Rajsekar Manokaran. Towards a characterization of constant-factor approximable finite-valued CSPs. *J. Comput. Syst. Sci.*, 97:14–27, 2018. doi:10.1016/j.jcss.2018.03.003.
- 14 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets weisfeiler and leman. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPICs*, pages 40:1–40:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ICALP.2018.40.
- 15 Tomás Feder and Moshe Y Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.
- 16 Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed constraint optimization problems and applications: A survey. *J. Artif. Int. Res.*, 61(1):623–698, 2018.
- 17 Konstantinos Georgiou, Avner Magen, and Madhur Tulsiani. Optimal sherali-adams gaps from pairwise independence. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2009. doi:10.1007/978-3-642-03685-9_10.
- 18 Mrinalkanti Ghosh and Madhur Tulsiani. From weak to strong linear programming gaps for all constraint satisfaction problems. *Theory Comput.*, 14(1):1–33, 2018. doi:10.4086/toc.2018.v014a010.
- 19 Martin Grohe, Kristian Kersting, Martin Mladenov, and Erkal Selman. Dimension reduction via colour refinement. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 505–516. Springer, 2014. doi:10.1007/978-3-662-44777-2_42.
- 20 Martin Grohe and Martin Otto. Pebble games and linear equations. *J. Symb. Log.*, 80(3):797–844, 2015. doi:10.1017/jsl.2015.28.
- 21 Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.
- 22 Neil Immerman and Eric Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity Theory Retrospective: In Honor of Juris Hartmanis on the Occasion of His Sixtieth Birthday*, pages 59–81, 1990.
- 23 Amit Kumar, Rajsekar Manokaran, Madhur Tulsiani, and Nisheeth K. Vishnoi. On lp-based approximability for strict csp. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1560–1573. SIAM, 2011. doi:10.1137/1.9781611973082.121.

- 24 Gabor Kun, Ryan O'Donnell, Suguru Tamaki, Yuichi Yoshida, and Yuan Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12*, page 484–495, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2090236.2090274.
- 25 Benoît Larose, Cynthia Loten, and Claude Tardif. A characterisation of first-order constraint satisfaction problems. *Log. Methods Comput. Sci.*, 3(4), 2007. doi:10.2168/LMCS-3(4:6)2007.
- 26 AA Leman and B Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.
- 27 László Lovász and Alexander Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim.*, 1(2):166–190, 1991. doi:10.1137/0801013.
- 28 Peter Mankin. Sherali–adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, 12:73–97, 2014.
- 29 Jaroslav Nešetřil and Vojtěch Rödl. Chromatically optimal rigid graphs. *Journal of Combinatorial Theory, Series B*, 46(2):133–141, 1989. doi:10.1016/0095-8956(89)90039-7.
- 30 Motakuri V. Ramana, Edward R. Scheinerman, and Daniel Ullman. Fractional isomorphism of graphs. *Discrete Mathematics*, 132(1):247–265, 1994. doi:10.1016/0012-365X(94)90241-0.
- 31 Edward R Scheinerman and Daniel H Ullman. *Fractional graph theory: a rational approach to the theory of graphs*. Courier Corporation, 2011.
- 32 Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discret. Math.*, 3(3):411–430, 1990. doi:10.1137/0403036.
- 33 Johan Thapper and Stanislav Zivný. The power of sherali-adams relaxations for general-valued csps. *SIAM J. Comput.*, 46(4):1241–1279, 2017. doi:10.1137/16M1079245.
- 34 Gottfried Tinhofer. Graph isomorphism and theorems of birkhoff type. *Computing*, 36(4):285–300, 1986. doi:10.1007/BF02240204.
- 35 Gottfried Tinhofer. A note on compact graphs. *Discret. Appl. Math.*, 30(2-3):253–264, 1991. doi:10.1016/0166-218X(91)90049-3.
- 36 Yuichi Yoshida and Yuan Zhou. Approximation schemes via sherali-adams hierarchy for dense constraint satisfaction problems and assignment problems. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 423–438. ACM, 2014. doi:10.1145/2554797.2554836.
- 37 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 331–342, October 2017. doi:10.1109/FOCS.2017.38.

A Applying the SA method to the Homomorphism problem

Here we shall show that the family of relaxations SA^k considered in the present paper is closely interleaved with the system of relaxations obtained by applying the SA method to a natural choice of initial polytope P .

Let \mathbf{A} and \mathbf{B} be σ -structures. We define polytope $P = P(\mathbf{A}, \mathbf{B})$ using a system of inequalities. The variables of the system are $x_{a,b}$ for each $a \in A$ and $b \in B$. Each variable must take a value in the range $[0, 1]$. We remark that by fixing some arbitrary ordering on the variables in $x_{a,b}$ we can represent any assignment on the variables $x_{a,b}$ with a tuple $\mathbf{x} \in \mathbb{R}^n$ with $n = |A| \cdot |B|$. Therefore we shall abuse notation and use $\mathbf{x}_{a,b}$ to refer to the value in \mathbf{x} corresponding to variable $x_{a,b}$.

The variables are subject to the following inequalities.

$$\sum_{b \in B} x_{a,b} = 1 \text{ for every } a \in A, \tag{1}$$

$$\sum_{\mathbf{a} \in \{\mathbf{a}\}} x_{\mathbf{a}, f(\mathbf{a})} \leq |\{\mathbf{a}\}| - 1 \quad \begin{array}{l} \text{for each } R \in \sigma, \mathbf{a} \in R(\mathbf{A}), \\ \text{and } f : \{\mathbf{a}\} \rightarrow B \text{ with } f(\mathbf{a}) \notin R(\mathbf{B}). \end{array} \quad (2)$$

Note that if h is a homomorphism from \mathbf{A} to \mathbf{B} then the assignment setting $x_{\mathbf{a}, h(\mathbf{a})} = 1$ for every $\mathbf{a} \in A$ and the rest of variables to zero is feasible.

Now let $P^k, k \geq 1$ be the sequence of polytopes obtained using the SA method. The next lemma shows that the sequence of relaxations defined by SA^k and P^k are interleaved.

► **Lemma 11.** *Let $k \geq 1$ and let r be the maximum arity of a relation in σ . Then*

1. *If $P^k \neq \emptyset$ and $r \leq k$ then SA^k is feasible.*
2. *If SA^{k+r-1} is feasible then $P^k \neq \emptyset$*

Proof. (1). Assume that $P^k \neq \emptyset$ and let \mathbf{y} be a feasible solution of P_L^k . We shall construct a feasible solution of SA^k . First, set every variable of the form $p_V(f)$ to y_K where $K = \{(a, f(a)) \mid a \in V\}$. We first observe that this assignment satisfies (SA1) and (SA2). Indeed, let $U \subseteq A$ with $|U| < k$, let $f : U \rightarrow B$, and let $I = \{(u, f(u)) : u \in U\}$. Then, multiplying the equality (1) with $a \in A \setminus U$ by $\prod_{i \in I} x_i$ and linearizing we obtain equality (SA2) for U, f , and $V = U \cup \{a\}$. In this way we can obtain all equalities in (SA2) for $|U| + 1 = |V|$. We note here that the rest of equalities in (SA2) along all equalities in (SA1) can be obtained as a linear combination.

Secondly, let us set the rest of variables. For every $(\mathbf{a}, R) \in \mathcal{C}_{\mathbf{A}}$ and $f : \{\mathbf{a}\} \rightarrow B$, set $p_{(\mathbf{a}, R)}(f)$ to be y_K where $K = \{(a, f(a)) \mid a \in \{\mathbf{a}\}\}$ (note that we are using implicitly the fact that $r \leq k$). Then, (SA3) follows directly from (SA2). Finally, it only remains to show that (SA4) is also satisfied. Indeed, for every $f(\mathbf{a}) \notin R(\mathbf{B})$ we obtain equality $p_{(\mathbf{a}, R)}(f) = 0$ if we multiply (2) by the term $\prod_{i \in K}$ and linearize. We want to note that, in fact, (1) also holds under the weaker assumption $r \leq k + 1$, but the proof is slightly more involved.

(2). Assume that SA^{k+r-1} is feasible. We construct a feasible solution \mathbf{y} of P_L^k as follows. For every $K \subseteq A \times B$ which satisfies $K = \{(a, f(a)) \mid a \in U\}$ for some $U \subseteq A$ with $|U| \leq k$ and $f : U \rightarrow B$, we set $y_K := p_U(f)$. Otherwise, we set y_K to zero.

Let us show that this assignment satisfies all inequalities in P_L^k . Let

$$\mathbf{c}^T \mathbf{y} \leq \mathbf{d} \quad (3)$$

be any inequality defining P_L^k . Since (3) is obtained by multiplying an inequality which contains at most r variables by a term of at most $k - 1$ variables, there exists a set $V \subseteq A$ with $|V| \leq r + k - 1$ such that for every variable y_K appearing in (3), V satisfies $K \subseteq V \times B$. Note that, by (SA1), variables $p_V(g), g : V \rightarrow B$ define a probability distribution. For every $g : V \rightarrow B$ in the support of this distribution, consider the assignment \mathbf{x}^g that sets $\mathbf{x}_{v,b}^g = 1$ if $v \in V$ and $b = g(v)$ and $x_{v,b} = 0$ otherwise.

Inequality (3) has been obtained by multiplying an inequality from (1) or (2) by a term and linearizing. We claim that in both cases, the inequality that has generated (3) is satisfied by \mathbf{x}^g . If the inequality generating (3) is $\sum_{b \in B} x_{a,b} = 1$ for some $a \in A$ this follows simply from the fact that $a \in V$. Assume now that (3) has been generated by inequality $\sum_{\mathbf{a} \in \{\mathbf{a}\}} x_{\mathbf{a}, f(\mathbf{a})} \leq |\{\mathbf{a}\}| - 1$. In this case note that $\{\mathbf{a}\} \subseteq V$ and then the claim follows from (SA3) and (SA4). This finalizes the proof of the claim.

Consequently, since \mathbf{x}^g is integral it follows that the assignment \mathbf{y}^g defined as $\mathbf{y}_K^g = \prod_{i \in K} \mathbf{x}_i^g$ satisfies (3). Finally, note that if we set $\alpha^g = p_V(g)$, then for every $K \subseteq V \times B$, \mathbf{y}_K is precisely given by the convex combination $\sum_g \alpha^g \mathbf{y}_K^g$. ◀

B Proof of Lemma 1

► **Lemma 1.** *Let \mathbf{A}, \mathbf{B} be σ -structures. Then $SA^k(\mathbf{A}, \mathbf{B})$ is feasible if and only if $SA^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$ is feasible.*

Proof (Sketch). The proof is purely syntactical although it is convenient to massage first a bit the LP formulations $SA^k(\mathbf{A}, \mathbf{B})$ and $SA^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$. We shall refer to the solutions of $SA^k(\mathbf{A}, \mathbf{B})$ and $SA^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$ by appropriately indexed sets of variables p, q respectively.

- In $SA^k(\mathbf{A}, \mathbf{B})$, it follows from (SA4) that we can safely replace all variables $p_{R(\mathbf{a})}(f)$ with $f(\mathbf{a}) \notin R(\mathbf{B})$ by 0.
- In $SA^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$ we are required a bit more of work. First, for each $j \leq k$ and each $\mathbf{a} \in A^j$, it follows from conditions (SA3) and (SA4) for $T_{j,S}$ ($S \subseteq [j]$) that for every $x \in B_k^*$, $q_{\mathbf{a}}(x)$ must take value 0 unless $x = f(\mathbf{a})$ for some function $f : \{\mathbf{a}\} \rightarrow B$. Hence, in a first stage we set $q_{\mathbf{a}}(x)$ to zero for each $j \leq k$, each $\mathbf{a} \in A^j$ and each x that is not a tuple of the form $f(\mathbf{a})$ for some function $f : \{\mathbf{a}\} \rightarrow B$.

Furthermore, it follows from condition (SA3) for $T_{j,i}$ that $q_{\mathbf{a}}(f(\mathbf{a})) = q_{\mathbf{a}'}(f(\mathbf{a}'))$ for every \mathbf{a}, \mathbf{a}' satisfying $\{\mathbf{a}\} = \{\mathbf{a}'\}$ and every $f : \{\mathbf{a}\} \rightarrow B$. Hence, in a second stage, for each $V \subseteq A$ with $|V| \leq k$ and every $f : V \rightarrow B$ we identify all variables $q_{\mathbf{a}}(f(\mathbf{a}))$ which satisfy $\{\mathbf{a}\} = V$.

Then, consider now the variables of the form $q_{R(\mathbf{a})}(x)$, $x \in B_k^*$. It follows from conditions (SA3) and (SA4) for R_S ($S \subseteq [\text{arity}(R)]$) that $q_{R(\mathbf{a})}(x)$ must be set to 0 unless $x = R(f(\mathbf{a}))$ for some function $f : \{\mathbf{a}\} \rightarrow B$.

The other variables in $SA^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$ are of the form $q_C(f)$ where $C \in \mathcal{C}_{\mathbf{A}_k^*}$. As we shall see they can always safely be identified with some of the other variables. Let us start first with the case in which C is a unary constraint. If $C = T_{j,S}(\mathbf{a})$ or $C = R_S(\mathbf{a})$, then it follows from (SA3) that $q_C(f) = q_{\mathbf{a}}(f(\mathbf{a}))$. Assume now that C is a binary constraint, that is $C = T_{j,i}(\mathbf{a}, \pi_i \mathbf{a})$ or $C = R_i(\mathbf{a}, \pi_i \mathbf{a})$. It follows again from (SA3) that $q_C(f) = q_{\mathbf{a}}(f(\mathbf{a}))$

Now we are ready to prove the lemma. In particular, consider the following one-to-one correspondence between the assignments in $SA^k(\mathbf{A}, \mathbf{B})$ and $SA^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$:

- Every variable $p_V(f)$ in $SA^k(\mathbf{A}, \mathbf{B})$ is assigned as variable $q_{\mathbf{a}}(f(\mathbf{a}))$ in $SA^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$ where \mathbf{a} is any tuple satisfying $\{\mathbf{a}\} = V$.
- Every variable $p_{R(\mathbf{a})}(f)$ in $SA^k(\mathbf{A}, \mathbf{B})$ is assigned as variable $q_{R(\mathbf{a})}(R(f(\mathbf{a})))$ in $SA^1(\mathbf{A}_k^*, \mathbf{B}_k^*)$.

It is not difficult to see that this correspondence preserves feasibility. ◀

C Proof of Lemma 3

► **Lemma 3.** *Let r be the maximum arity among all relations in σ and assume that $r \leq k$. Then for every pair of structures \mathbf{A}, \mathbf{B} the following are equivalent:*

1. $\mathbf{A} \equiv_k \mathbf{B}$
2. $\text{Hom}(\mathbf{Q}; \mathbf{A}) = \text{Hom}(\mathbf{Q}; \mathbf{B})$ for every structure \mathbf{Q} of treewidth $< k$.

From Theorem 2 it follows that for a pair of σ -structures \mathbf{A} and \mathbf{B} , $\mathbf{A}_k^* \equiv_k \mathbf{B}_k^*$ if and only if $\text{Hom}(\mathbf{T}, \mathbf{A}_k^*) = \text{Hom}(\mathbf{T}, \mathbf{B}_k^*)$ for every σ_k^* -ftree \mathbf{T} . So it only remains to prove the following.

▷ **Claim 12.** Assume that $r \leq k$. Then the following are equivalent:

1. $\text{Hom}(\mathbf{Q}, \mathbf{A}) = \text{Hom}(\mathbf{Q}, \mathbf{B})$ for every σ -structure \mathbf{Q} of treewidth $< k$;
2. $\text{Hom}(\mathbf{T}, \mathbf{A}_k^*) = \text{Hom}(\mathbf{T}, \mathbf{B}_k^*)$ for every σ_k^* -ftree \mathbf{T} .

Proof. (1) \Rightarrow (2). Let \mathbf{T} be a σ^* -ftree. It follows immediately that if (1) holds then both \mathbf{A} and \mathbf{B} must have the same number of elements and constraints. It then follows that (2) holds for \mathbf{T} if it consists of a single element and no constraints at all. Consequently we can safely assume that all elements in \mathbf{T} participate in at least one constraint.

In what follows $\mathbf{D}_k^* \in \{\mathbf{A}_k^*, \mathbf{B}_k^*\}$. Let t be any node in T . Since t participates in a constraint it follows that the possible image of t in homomorphism from \mathbf{T} is heavily restricted. In particular, if the image of t according to some homomorphism from \mathbf{T} to \mathbf{D}_k^* is in D^j for some $j \leq k$ then necessarily the image of t in any homomorphism from \mathbf{T} to any structure $\mathbf{C}_k^* \in \{\mathbf{A}_k^*, \mathbf{B}_k^*\}$ must be in C^j . This means that we can safely add constraint $T_{j,\emptyset}(t)$ to \mathbf{T} without altering $\text{Hom}(\mathbf{T}, \mathbf{A}_k^*)$ or $\text{Hom}(\mathbf{T}, \mathbf{B}_k^*)$.

Likewise, if some homomorphism from \mathbf{T} to a structure \mathbf{D}_k^* maps t to a constraint $R(\mathbf{d})$, then likewise we can assume that constraint $R_\emptyset(t)$ belongs to \mathbf{T} .

To complete the proof we shall show that it is always possible to construct from \mathbf{T} a σ -structure \mathbf{Q} of treewidth $< k$ such that $|\text{Hom}(\mathbf{T}, \mathbf{D}_k^*)| = |\text{Hom}(\mathbf{Q}, \mathbf{D})|$. It is convenient to construct \mathbf{Q} in two stages. First, let us construct a σ -structure \mathbf{P} (not necessarily of treewidth $< k$) satisfying that $|\text{Hom}(\mathbf{T}, \mathbf{D}_k^*)| = |\text{Hom}(\mathbf{P}, \mathbf{D})|$. We shall allow to use equalities in \mathbf{P} , i.e., constraints of the form $p_1 = p_2$, indicating that p_1 and p_2 must be assigned to the same element in D .

We shall define \mathbf{P} along with a function α mapping every element t of \mathbf{T} to a j -ary tuple of elements in \mathbf{P} ($j \leq k$) inductively on the number of elements of \mathbf{T} as follows.

Assume (base case) that \mathbf{T} contains a unique element t . As discussed above we can assume that \mathbf{T} contains constraint $T_{j,\emptyset}(t)$ for some $j \leq k$ or $R_\emptyset(t)$ for some $R \in \sigma$. In the first case, we set the universe of \mathbf{P} to contain j new elements p_1, \dots, p_j . Furthermore, for every unary constraint $T_{j,S}(t)$ in \mathbf{T} and every $i, i' \in S$, we include in \mathbf{P} the equality $p_i = p_{i'}$, and we define $\alpha(t) = (p_1, \dots, p_j)$. In the second case, we set the universe of \mathbf{P} to contain $\text{arity}(R)$ new elements $p_1, \dots, p_{\text{arity}(R)}$ and we include in \mathbf{P} the constraint $R(p_1, \dots, p_{\text{arity}(R)})$. Similarly to the previous case, for every unary constraint $R_S(t)$ in \mathbf{T} and every $i, i' \in S$, we include in \mathbf{P} the equality $p_i = p_{i'}$. Finally, we set $\alpha(t) = (p_1, \dots, p_{\text{arity}(R)})$.

Let us consider now the inductive case. Let t_1 and t_2 be nodes that participate in a binary constraint $U(t_1, t_2)$ (recall that U is either $T_{j,\mathbf{i}}$ or $R_{\mathbf{i}}$) in \mathbf{T} . By removing this constraint \mathbf{T} gets divided in two ftrees \mathbf{T}_1 and \mathbf{T}_2 such that \mathbf{T}_1 contains t_1 and \mathbf{T}_2 contains t_2 . Now, assume that \mathbf{P}_i and α_i are already constructed for \mathbf{T}_i , $i = 1, 2$. We are ready to define \mathbf{P} . First, we compute the disjoint union of \mathbf{P}_1 and \mathbf{P}_2 . Then, we add some further equalities depending on constraint $U(t_1, t_2)$. Consider first the case that $U = T_{j_1,\mathbf{i}}$, $\mathbf{i} = (i_1, \dots, i_{j_2}) \in [j_1]^{j_2}$ for some $j_1, j_2 \leq k$ and let $\alpha_i(t_i) = (p_{i_1}^{i_1}, \dots, p_{i_{j_2}}^{i_{j_2}})$, $i = 1, 2$. Then, for every $\ell \leq j_2$ we add the equality $p_\ell^2 = p_\ell^1$. Finally, for every $t \in T$ we define $\alpha(t)$ to be $\alpha_i(t)$ where \mathbf{T}_i contains t . The procedure is identical for $U = R_{\mathbf{i}}$, where we just substitute j_1 by $\text{arity}(R)$. It follows immediately from the definition that $\text{Hom}(\mathbf{P}, \mathbf{D}) = \text{Hom}(\mathbf{T}, \mathbf{D}_k^*)$.

Finally, let us define \mathbf{Q} to be the structure obtained by identifying (i.e., merging into a single element) all elements in \mathbf{P} joined by a chain of equalities. It is immediate that $\text{Hom}(\mathbf{P}, \mathbf{D}) = \text{Hom}(\mathbf{Q}, \mathbf{D})$.

We shall conclude by giving a tree-decomposition (G, β) of \mathbf{Q} of width $< k$. In particular, let G be the tree where the vertex set is precisely the universe of \mathbf{T} and two different nodes are adjacent if both participate in some common constraint in \mathbf{T} and let $\beta(t) = \{\alpha(t)\}$.

(2) \Rightarrow (1). Let \mathbf{Q} be a σ -structure of treewidth $< k$ and let $\mathbf{D} \in \{\mathbf{A}, \mathbf{B}\}$. We note here that we can assume that \mathbf{Q} is connected since if \mathbf{Q} is the disjoint union of structures \mathbf{Q}_1 and \mathbf{Q}_2 then $\text{Hom}(\mathbf{Q}, \mathbf{D}) = \text{Hom}(\mathbf{Q}_1, \mathbf{D}) \cdot \text{Hom}(\mathbf{Q}_2, \mathbf{D})$. We shall show that there exists a σ_k^* -ftree \mathbf{T} such that $\text{Hom}(\mathbf{T}, \mathbf{D}_k^*) = \text{Hom}(\mathbf{Q}, \mathbf{D})$. Let (G, β) be a tree-decomposition of

width at most k of \mathbf{Q} . It is well-known and easy to prove that, since \mathbf{Q} is connected we can always construct (G, β) in such a way that for every pair u, v of adjacent nodes in G , $\beta(u) \subseteq \beta(v)$ or $\beta(v) \subseteq \beta(u)$. Furthermore, it is easy to see that we can further enforce that for every constraint $R(\mathbf{q})$ in \mathbf{Q} there exists a node $v \in G$ such that $\beta(v) = \{\mathbf{q}\}$.

The universe T of \mathbf{T} is $V \cup \mathcal{C}_{\mathbf{Q}}$ where V is the node-set of G . Furthermore \mathbf{T} contains the following constraints.

Let us start with the unary constraints. Let t be an element in \mathbf{T} . If $t = v \in G$ then we include in \mathbf{T} a constraint $T_{j_v, \emptyset}(t)$ where $j_v = |\beta(v)|$. Otherwise, if $t = R(\mathbf{q}) \in \mathcal{C}_{\mathbf{Q}}$ then we include in \mathbf{T} all constraints $R_{\{i, i'\}}(t)$ where $\mathbf{q}[i] = \mathbf{q}[i']$.

Now, let us turn our attention to the binary constraints. Fix some arbitrary ordering on Q and for every $v \in V$ let $\mathbf{q}^v = (q_1^v, \dots, q_{j_v}^v)$ be an array containing the nodes in $\beta(v)$ following this fixed order.

Then, for every edge (u, v) in G include constraint $T_{j_u, \mathbf{i}}(u, v)$ where $\mathbf{i} = (i_1, \dots, i_{j_v})$ is defined as follows. First, we assume without loss of generality that $\beta(v) \subseteq \beta(u)$. Then, for every $\ell \leq j_v$, i_ℓ is defined to be such that $\mathbf{q}^u[i_\ell] = \mathbf{q}^v[\ell]$.

Finally, for every constraint $t = R(\mathbf{q})$ in \mathbf{Q} we pick some element $v \in V$ satisfying $\{\mathbf{q}\} = \beta(v)$ and we add the constraint $R_{\mathbf{i}}(t, v)$ with $\mathbf{i} = (i_1, \dots, i_{j_v})$ where i_ℓ satisfies $\mathbf{q}[i_\ell] = \mathbf{q}^v[\ell]$. It is immediate to see that \mathbf{T} is an ftree and that $\text{Hom}(\mathbf{Q}, \mathbf{D}) = \text{Hom}(\mathbf{T}, \mathbf{D}_k^*) \triangleleft$

A Decidable Equivalence for a Turing-Complete, Distributed Model of Computation

Arnaldo Cesco  

Department of Computer Science and Engineering, University of Bologna, Italy

Roberto Gorrieri  

Department of Computer Science and Engineering, University of Bologna, Italy

Abstract

Place/Transition Petri nets with inhibitor arcs (PTI nets for short), which are a well-known Turing-complete, distributed model of computation, are equipped with a decidable, behavioral equivalence, called *pti-place bisimilarity*, that conservatively extends place bisimilarity defined over Place/Transition nets (without inhibitor arcs). We prove that *pti-place bisimilarity* is sensible, as it respects the causal semantics of PTI nets.

2012 ACM Subject Classification Theory of computation → Distributed computing models

Keywords and phrases Petri nets, Inhibitor arc, Behavioral equivalence, Bisimulation, Decidability

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.28

Related Version *Previous Version*: <https://arxiv.org/abs/2104.14859> [11]

1 Introduction

Place/Transition Petri nets with inhibitor arcs (PTI nets, for short), originally introduced in [2], are a well-known (see, e.g., [7, 19, 26]), Turing-complete (as proved first by Agerwala in [1]), distributed model of computation, largely exploited, e.g., for modeling systems with priorities [17], for performance evaluation of distributed systems [3] and to provide π -calculus [24, 28] with a net semantics [8].

As finite PTI nets constitute a Turing-complete model of computation, essentially all the properties of interest are undecidable, notably the reachability problem, and so even termination: it is undecidable whether a deadlock marking is reachable from the initial one. Also interleaving bisimulation equivalence is undecidable for finite PTI nets, as it is already undecidable [20] on the subclass of finite P/T nets [27]. Similarly, one can prove that also well-known truly-concurrent behavioral equivalences, such as *fully-concurrent* bisimilarity [6], are undecidable [12] for finite PTI nets. Despite this, we show that it is possible to define a *sensible*, behavioral equivalence which is actually *decidable* on finite PTI nets. This equivalence, we call *pti-place bisimilarity*, is a conservative extension of *place bisimilarity on finite P/T nets*, introduced in [4] as an improvement of *strong bisimulation* [25], (a relation proposed by Olderog in [25] on safe nets which fails to induce an equivalence relation), and recently proved decidable in [16].

Place bisimilarity on finite P/T nets is an equivalence over markings, based on relations over the *finite set of net places*, rather than over the (possibly infinite) set of net markings. This equivalence is very natural and intuitive: as a place can be interpreted as a sequential process type (and each token in this place as an instance of a sequential process of that type), a place bisimulation states which kinds of sequential processes (composing the distributed system represented by the finite P/T net) are to be considered as equivalent. Moreover, this equivalence does respect the causal behavior of P/T nets, as van Glabbeek proved in [29] that it is slightly finer than *structure preserving bisimilarity* [29], in turn slightly finer than *fully-concurrent bisimilarity* [6].



© Arnaldo Cesco and Roberto Gorrieri;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 28; pp. 28:1–28:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We extend this idea in order to be applicable to PTI nets. Informally, a binary relation R over the set S of places is a *pti-place bisimulation* if for all markings m_1 and m_2 which are *bijectively* related via R (denoted by $(m_1, m_2) \in R^\oplus$, where R^\oplus is called the *additive closure* of R), if m_1 can perform transition t_1 , reaching marking m'_1 , then m_2 can perform a transition t_2 , reaching m'_2 , such that

- the pre-sets of t_1 and t_2 are related by R^\oplus , the label of t_1 and t_2 is the same, the post-sets of t_1 and t_2 are related by R^\oplus , and also $(m'_1, m'_2) \in R^\oplus$, as required by a place bisimulation [4, 16], but additionally it is required that
- whenever $(s, s') \in R$, s belongs to the inhibiting set of t_1 if and only if s' belongs to the inhibiting set of t_2 ;

and symmetrically if m_2 moves first. Two markings m_1 and m_2 are *pti-place bisimilar*, denoted by $m_1 \sim_p m_2$, if a pti-place bisimulation R exists such that $(m_1, m_2) \in R^\oplus$.

We prove that pti-place bisimilarity is an equivalence, but it is not coinductive as the union of pti-place bisimulations may be not a pti-place bisimulation; so, in general, there is not a largest pti-place bisimulation, rather many maximal pti-place bisimulations. In fact, pti-place bisimilarity is the relation on markings given by the union of the additive closure of each maximal pti-place bisimulation. We also prove that \sim_p is sensible, as it respects the causal semantics of PTI nets. As a matter of fact, following the approach in [9, 10], we define a novel, process-oriented, bisimulation-based, behavioral semantics for PTI nets, called *causal-net bisimilarity*, and we prove that this is slightly coarser than pti-place bisimilarity.

The other main contribution of this paper is to show that \sim_p is decidable for finite PTI nets. As a place relation $R \subseteq S \times S$ is finite if the set S of places is finite, there are finitely many place relations for a finite net. We can list all these place relations, say R_1, R_2, \dots, R_n . It is possible to decide whether R_i is a pti-place bisimulation by checking two *finite* conditions over a *finite* number of marking pairs: this is a non-obvious observation, as a pti-place bisimulation requires that the pti-place bisimulation conditions hold for the infinitely many pairs (m_1, m_2) belonging to R_i^\oplus . Hence, to decide whether $m_1 \sim_p m_2$, it is enough to check, for $i = 1, \dots, n$, whether R_i is a pti-place bisimulation and, in such a case, whether $(m_1, m_2) \in R_i^\oplus$.

The paper is organized as follows. Section 2 recalls the basic definitions about PTI nets, including their causal semantics. Section 3 deals with pti-place bisimilarity, shows that it is an equivalence relation, that it is not coinductive, and that it is slightly finer than causal-net bisimilarity. Section 4 shows that \sim_p is decidable. Finally, Section 5 discusses some related literature and future research.

2 Basic definitions about P/T nets and PTI nets

► **Definition 1 (Multiset).** Let \mathbb{N} be the set of natural numbers. Given a finite set S , a multiset over S is a function $m : S \rightarrow \mathbb{N}$. The support set $\text{dom}(m)$ of m is $\{s \in S \mid m(s) \neq 0\}$. The set of all multisets over S , denoted by $\mathcal{M}(S)$, is ranged over by m . We write $s \in m$ if $m(s) > 0$. The multiplicity of s in m is given by the number $m(s)$. The size of m , denoted by $|m|$, is the number $\sum_{s \in S} m(s)$, i.e., the total number of its elements. A multiset m such that $\text{dom}(m) = \emptyset$ is called empty and is denoted by θ . We write $m \subseteq m'$ if $m(s) \leq m'(s)$ for all $s \in S$. Multiset union $_ \oplus _$ is defined as follows: $(m \oplus m')(s) = m(s) + m'(s)$. Multiset difference $_ \ominus _$ is defined as follows: $(m_1 \ominus m_2)(s) = \max\{m_1(s) - m_2(s), 0\}$. The scalar product of a number j with m is the multiset $j \cdot m$ defined as $(j \cdot m)(s) = j \cdot (m(s))$. By s_i we also denote the multiset with s_i as its only element. Hence, a multiset m over $S = \{s_1, \dots, s_n\}$ can be represented as $k_1 \cdot s_1 \oplus k_2 \cdot s_2 \oplus \dots \oplus k_n \cdot s_n$, where $k_j = m(s_j) \geq 0$ for $j = 1, \dots, n$.

► **Definition 2** (Place/Transition Petri net). A labeled, finite Place/Transition Petri net (P/T net for short) is a tuple $N = (S, A, T)$, where

- S is the finite set of places, ranged over by s (possibly indexed),
- A is the finite set of labels, ranged over by ℓ (possibly indexed), and
- $T \subseteq (\mathcal{M}(S) \setminus \{\emptyset\}) \times A \times (\mathcal{M}(S) \setminus \{\emptyset\})$ is the finite set of transitions, ranged over by t (possibly indexed).

Given a transition $t = (m, \ell, m')$, we use the notation:

- $\bullet t$ to denote its pre-set m (which cannot be empty) of tokens to be consumed;
- $l(t)$ for its label ℓ , and
- t^\bullet to denote its post-set m' (which cannot be an empty multiset) of tokens to be produced.

Hence, transition t can be also represented as $\bullet t \xrightarrow{l(t)} t^\bullet$. We also define the flow function $\text{flow} : (S \times T) \cup (T \times S) \rightarrow \mathbb{N}$ as follows: for all $s \in S$, for all $t \in T$, $\text{flow}(s, t) = \bullet t(s)$ and $\text{flow}(t, s) = t^\bullet(s)$. We will use F to denote the flow relation $\{(x, y) \in (S \times T) \cup (T \times S) \mid \text{flow}(x, y) > 0\}$. Finally, we define pre-sets and post-sets also for places as follows: $\bullet s = \{t \in T \mid s \in \bullet t\}$ and $s^\bullet = \{t \in T \mid s \in t^\bullet\}$. Note that while the pre-set (post-set) of a transition is, in general, a multiset, the pre-set (post-set) of a place is a set.

► **Definition 3** (Place/Transition net with inhibitor arcs). A finite Place/Transition net with inhibitor arcs (PTI net for short) is a tuple $N = (S, A, T, I)$, where

- (S, A, T) is a finite P/T net;
- $I \subseteq S \times T$ is the inhibiting relation.

Given a transition $t \in T$, we denote by ${}^\circ t$ its inhibiting set $\{s \in S \mid (s, t) \in I\}$ of places to be tested for absence of tokens. Hence, a transition t can be also represented as $(\bullet t, {}^\circ t) \xrightarrow{l(t)} t^\bullet$.

We use the standard graphical convention for Petri nets. In particular, a pair (s, t) in the inhibiting relation I is graphically represented by an arc from s to t ending with a small circle on the transition side.

► **Definition 4** (Marking, PTI net system). A PTI net system $N(m_0)$ is a tuple (S, A, T, I, m_0) , where (S, A, T, I) is a PTI net and m_0 is a multiset over S , called the initial marking. We also say that $N(m_0)$ is a marked net.

► **Definition 5** (Token game). A transition t is enabled at m , denoted $m[t]$, if $\bullet t \subseteq m$ and ${}^\circ t \cap \text{dom}(m) = \emptyset$. The execution, or firing, of t enabled at m produces the marking $m' = (m \ominus \bullet t) \oplus t^\bullet$, written $m[t]m'$.

► **Definition 6** (Firing sequence, reachable marking, safe net). A firing sequence starting at m is defined inductively as follows:

- $m[\epsilon]m$ is a firing sequence (where ϵ denotes an empty sequence of transitions) and
- if $m[\sigma]m'$ is a firing sequence and $m'[t]m''$, then $m[\sigma t]m''$ is a firing sequence.

The set of reachable markings from m is $[m] = \{m' \mid \exists \sigma. m[\sigma]m'\}$. A PTI system $N = (S, A, T, I, m_0)$ is safe if for each marking $m \in [m_0]$, we have that $m(s) \leq 1$ for all $s \in S$.

2.1 Causal semantics for P/T nets and PTI nets

We outline some definitions about the causal semantics of P/T nets, adapted from the literature (cf., e.g., [6, 29, 13, 25]).

► **Definition 7** (Acyclic net). A P/T net $N = (S, A, T)$ is acyclic if its flow relation F is acyclic (i.e., $\nexists x$ such that $x F^+ x$, where F^+ is the transitive closure of F).

The concurrent semantics of a marked P/T net is defined by a class of particular acyclic safe nets, where places are not branched (hence they represent a single run) and all arcs have weight 1. This kind of net is called *causal net*. We use the name C (possibly indexed) to denote a causal net, the set B to denote its places (called *conditions*), the set E to denote its transitions (called *events*), and L to denote its labels.

► **Definition 8** (Causal P/T net). *A causal net is a finite marked net $C(m_0) = (B, L, E, m_0)$ satisfying the following conditions:*

1. C is acyclic;
 2. $\forall b \in B \quad |\bullet b| \leq 1 \wedge |b^\bullet| \leq 1$ (i.e., the places are not branched);
 3. $\forall b \in B \quad m_0(b) = \begin{cases} 1 & \text{if } \bullet b = \emptyset \\ 0 & \text{otherwise;} \end{cases}$
 4. $\forall e \in E \quad \bullet e(b) \leq 1 \wedge e^\bullet(b) \leq 1$ for all $b \in B$ (i.e., all the arcs have weight 1).
- We denote by $\text{Min}(C)$ the set m_0 , and by $\text{Max}(C)$ the set $\{b \in B \mid b^\bullet = \emptyset\}$.

Note that any reachable marking of a causal net is a set, i.e., this net is *safe*; in fact, the initial marking is a set and, assuming by induction that a reachable marking m is a set and enables e , i.e., $m[e]m'$, then also $m' = (m \ominus \bullet e) \oplus e^\bullet$ is a set, as the net is acyclic and because of the condition on the shape of the post-set of e (weights can only be 1).

As the initial marking of a causal P/T net is fixed by its shape (according to item 3 of Definition 8), in the following, in order to make the notation lighter, we often omit the indication of the initial marking (also in their graphical representation), so that the causal net $C(m_0)$ is denoted by C .

► **Definition 9** (Moves of a causal P/T net). *Given two causal nets $C = (B, L, E, m_0)$ and $C' = (B', L, E', m_0)$, we say that C moves in one step to C' through e , denoted by $C[e]C'$, if $\bullet e \subseteq \text{Max}(C)$, $E' = E \cup \{e\}$ and $B' = B \cup e^\bullet$.*

► **Definition 10** (Folding and Process). *A folding from a causal P/T net $C = (B, L, E, m_0)$ into a P/T net system $N(m_0) = (S, A, T, m_0)$ is a function $\rho : B \cup E \rightarrow S \cup T$, which is type-preserving, i.e., such that $\rho(B) \subseteq S$ and $\rho(E) \subseteq T$, satisfying the following:*

- $L = A$ and $l(e) = l(\rho(e))$ for all $e \in E$;
- $\rho(m_0) = m_0$, i.e., $m_0(s) = |\rho^{-1}(s) \cap m_0|$;
- $\forall e \in E, \rho(\bullet e) = \bullet \rho(e)$, i.e., $\rho(\bullet e)(s) = |\rho^{-1}(s) \cap \bullet e|$ for all $s \in S$;
- $\forall e \in E, \rho(e^\bullet) = \rho(e)^\bullet$, i.e., $\rho(e^\bullet)(s) = |\rho^{-1}(s) \cap e^\bullet|$ for all $s \in S$.

A pair (C, ρ) , where C is a causal net and ρ a folding from C to a net system $N(m_0)$, is a process of $N(m_0)$.

► **Definition 11** (Moves of a P/T process). *Let $N(m_0) = (S, A, T, m_0)$ be a net system and let (C_i, ρ_i) , for $i = 1, 2$, be two processes of $N(m_0)$. We say that (C_1, ρ_1) moves in one step to (C_2, ρ_2) through e , denoted by $(C_1, \rho_1) \xrightarrow{e} (C_2, \rho_2)$, if $C_1[e]C_2$ and $\rho_1 \subseteq \rho_2$.*

Following [9, 10], we define here a possible causal semantics for PTI nets. In order to maintain the pleasant property that a process univocally determines the causal dependencies among its events, it is not enough to just enrich causal P/T nets with inhibitor arcs. Indeed, the reason why a condition is empty may influence the causal relation of events. To solve the problem, in [9, 10] inhibitor arcs are partitioned into two sets: *before* inhibitor arcs and *after* inhibitor arcs. If a condition is connected to an event by a before inhibitor arc, the event fires because the condition has not held yet; if they are connected by an after inhibitor arc, the event fires because the condition does not hold anymore.

► **Definition 12** (Causal PTI net). A causal PTI net is a tuple $C(m_0) = (B, L, E, Y^{be}, Y^{af}, m_0)$ satisfying the following conditions, denoting the flow relation of C by F :

1. (B, L, E, m_0) is a causal P/T net;
2. $(B, L, E, Y^{be} \cup Y^{af}, m_0)$ is a marked PTI net;
3. before and after requirements are met, i.e.
 - (a) If $b Y^{be} e$, then there exists $e' \in E$ such that $e' F b$, and
 - (b) If $b Y^{af} e$, then there exists $e' \in E$ such that $b F e'$;
4. relation $F \cup \prec_{af} \cup \prec_{be}$ is acyclic, where $\prec_{af} = F^{-1} \circ Y^{af}$ and $\prec_{be} = (Y^{be})^{-1} \circ F^{-1}$.

We denote by $Min(C)$ the set m_0 , and by $Max(C)$ the set $\{b \in B \mid b^\bullet = \emptyset\}$.

Relation $\prec_{af} \subseteq E \times E$ states that $e \prec_{af} e'$ if e consumes the token in a place b inhibiting e' : this is clearly a causal dependency. Instead, relation $\prec_{be} \subseteq E \times E$ states that $e \prec_{be} e'$ if e' produces a token in a place b inhibiting e : this is clearly a temporal precedence, because the two events can be causally independent, yet they cannot occur in any order, as if e' occurs, then e is disabled.

► **Definition 13** (Folding and PTI process). A folding from a causal PTI net $C = (B, L, E, Y^{be}, Y^{af}, m_0)$ into a PTI net system $N(m_0) = (S, A, T, I, m_0)$ is a function $\rho : B \cup E \rightarrow S \cup T$, which is type-preserving, i.e., such that $\rho(B) \subseteq S$ and $\rho(E) \subseteq T$, satisfying the following:

- ρ is a P/T folding from (B, L, E, m_0) into (S, A, T, m_0) ;
- for all $s \in S$ and $e \in E$, if $(s, \rho(e)) \in I$ then for all $b \in B$ such that $\rho(b) = s$, it holds $(b, e) \in Y^{be} \cup Y^{af} \cup F^{-1}$, and
for all $b \in B$ and $e \in E$, if $(b, e) \in Y^{be} \cup Y^{af}$ then $(\rho(b), \rho(e)) \in I$.

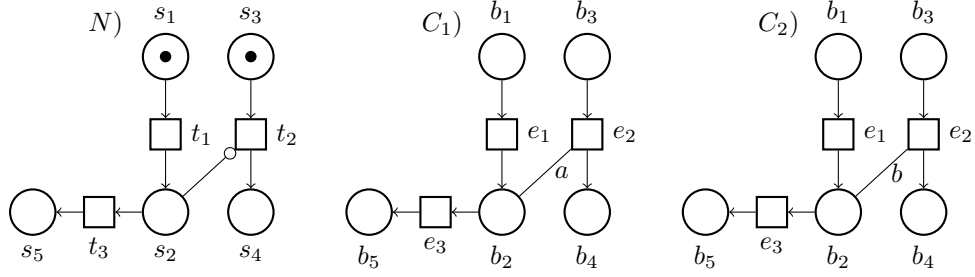
A pair (C, ρ) , where C is a causal PTI net and ρ a folding from C to a PTI net system $N(m_0)$, is a PTI process of $N(m_0)$.

Each inhibitor arc in the causal net has a corresponding inhibitor arc in the net system. The only case where a condition b is not connected by an inhibitor arc to an event e is when b is in the post-set of e : as b starts to hold only after e occurs, the only possibility is to put a before arc. This would make the relation \prec_{be} reflexive, invalidating item 4 of Definition 12. However, since b is in the post-set of e , we are sure that e happens before b is fulfilled, hence making useless the presence of a before inhibitor arc. For this reason, with the requirement $(b, e) \in Y^{be} \cup Y^{af} \cup F^{-1}$, we ask for the presence of an inhibitor arc only if there exists no flow from e to b .

► **Definition 14** (Moves of a PTI process). Let $N(m_0) = (S, A, T, I, m_0)$ be a PTI net system and let (C_i, ρ_i) , for $i = 1, 2$, be two PTI processes of $N(m_0)$, where $C_i = (B_i, L, E_i, Y_i^{be}, Y_i^{af}, m_0)$. We say that (C_1, ρ_1) moves in one step to (C_2, ρ_2) through e , denoted by $(C_1, \rho_1) \xrightarrow{e} (C_2, \rho_2)$, if the following hold:

- $e \subseteq Max(C_1)$, $E_2 = E_1 \cup \{e\}$, $B_2 = B_1 \cup e^\bullet$, $\rho_1 \subseteq \rho_2$, i.e. the P/T process of (C_1, ρ_1) moves in one step through e to the P/T process of (C_2, ρ_2) .
- Given two relations \mathcal{B} and \mathcal{A} , defined as
 - $\forall b \in e^\bullet, \forall e' \in E_1$ we have $b \mathcal{B} e'$ if and only if $(\rho_2(b), \rho_2(e')) \in I$,
 - $\forall b \in B_2$ such that $b^\bullet \neq \emptyset$, we have $b \mathcal{A} e$ if and only if $(\rho_2(b), \rho_2(e)) \in I$,
 we have $\{b \in B_2 \mid b \mathcal{A} e\} \cap Max(C_1) = \emptyset$.
- Finally, $Y_2^{be} = Y_1^{be} \cup \mathcal{B}$ and $Y_2^{af} = Y_1^{af} \cup \mathcal{A}$.

The item $\{b \in B_2 \mid b \mathcal{A} e\} \cap Max(C_1) = \emptyset$ models the fact that a transition can fire only if all its inhibiting places are free. Indeed, an event can fire only if its (so far known) inhibiting conditions are not maximal. Note that, by construction, before arcs can connect



■ **Figure 1** A marked PTI net and two PTI causal nets corresponding to its two maximal processes.

only new inhibiting conditions to past events and in particular we do not allow before arcs connecting a condition in the post-set of a newly added event e with the event e itself. Moreover, after arcs can only connect old inhibiting conditions to the new event e and since $\{b \in B_2 \mid b \mathcal{A} e\} \cap \text{Max}(C_1) = \emptyset$, the old inhibiting conditions cannot be in the pre-set of the newly added event e . Therefore, both relations \prec_2^{be} and \prec_2^{af} are acyclic, and since F_2 is acyclic too, (C_2, ρ_2) is truly a process of $N(m_0)$.

► **Example 15.** Consider the three nets in Figure 1, where we use the graphical convention that *before* inhibitor arcs and *after* inhibitor arcs are represented by lines between a condition and an event: the former labeled by b , the latter labeled by a . The initial marking of N is $m_0 = s_1 \oplus s_3$. The shape of a process generated by $N(m_0)$ may depend on the order of transitions in a given transition sequence. As a matter of fact, transition sequences containing the same transitions but in a different order may generate different processes, e.g. C_1 and C_2 . Indeed, C_1 represents the transition sequence $t_1 t_3 t_2$, while C_2 represents the transition sequence $t_2 t_1 t_3$. In the former case, t_1 fires first, so that t_2 can only fire after the inhibiting token in s_2 has been cleaned up by transition t_3 : therefore the causal net has an after arc between b_2 and e_2 . In the latter case, t_2 is the first transition to fire and there are no tokens in the inhibiting place b_2 , therefore the causal net has a before arc between b_2 and e_2 . Note that the underlying causal P/T net of these two processes is the same, but before and after inhibitor arcs are different.

We are now ready to introduce a novel behavioral relation for PTI nets, namely causal-net bisimulation, which is an interesting relation in its own right, as the induced equivalence, namely *causal-net bisimilarity*, on P/T nets coincides with *structure-preserving bisimilarity* [29], and so it is slightly finer than *fully-concurrent bisimilarity* [6]. However, since we conjecture that causal-net bisimilarity is undecidable (already on finite P/T nets), we will use this behavioral relation only for comparison with pti-place bisimilarity, showing the the latter is a finer, but decidable, approximation of the former.

► **Definition 16 (Causal-net bisimulation).** Let $N = (S, A, T, I)$ be a PTI net. A causal-net bisimulation is a relation R , composed of triples of the form (ρ_1, C, ρ_2) , where, for $i = 1, 2$, (C, ρ_i) is a process of $N(m_{0i})$ for some m_{0i} , such that if $(\rho_1, C, \rho_2) \in R$ then

- i) $\forall t_1, C', \rho'_1$ such that $(C, \rho_1) \xrightarrow{e} (C', \rho'_1)$, where $\rho'_1(e) = t_1$, $\exists t_2, \rho'_2$ such that $(C, \rho_2) \xrightarrow{e} (C', \rho'_2)$, where $\rho'_2(e) = t_2$, and $(\rho'_1, C', \rho'_2) \in R$;
- ii) symmetrically, $\forall t_2, C', \rho'_2$ such that $(C, \rho_2) \xrightarrow{e} (C', \rho'_2)$, where $\rho'_2(e) = t_2$, $\exists t_1, \rho'_1$ such that $(C, \rho_1) \xrightarrow{e} (C', \rho'_1)$, where $\rho'_1(e) = t_1$, and $(\rho'_1, C', \rho'_2) \in R$.

Two markings m_1 and m_2 of N are *cn-bisimilar*, denoted by $m_1 \sim_{cn} m_2$, if there exists a causal-net bisimulation R containing a triple $(\rho_1^0, C^0, \rho_2^0)$, where C^0 contains no events and $\rho_i^0(\text{Min}(C^0)) = \rho_i^0(\text{Max}(C^0)) = m_i$ for $i = 1, 2$.

If $m_1 \sim_{cn} m_2$, then these two markings have the same causal PTI nets, so that the executions originating from the two markings have the same causal dependencies (determined by F and \prec_{af}) and the same temporal dependencies (determined by \prec_{be}). Causal-net bisimilarity \sim_{cn} is an equivalence relation (see the preliminary version of this paper [11]).

3 Pti-place bisimilarity

We now present pti-place bisimilarity, which conservatively extends *place bisimilarity* [4, 16] to the case of PTI nets. First, an auxiliary definition.

3.1 Additive closure and its properties

► **Definition 17** (Additive closure). *Given a PTI net $N = (S, A, T, I)$ and a place relation $R \subseteq S \times S$, we define a marking relation $R^\oplus \subseteq \mathcal{M}(S) \times \mathcal{M}(S)$, called the additive closure of R , as the least relation induced by the following axiom and rule.*

$$\frac{}{(\theta, \theta) \in R^\oplus} \quad \frac{(s_1, s_2) \in R \quad (m_1, m_2) \in R^\oplus}{(s_1 \oplus m_1, s_2 \oplus m_2) \in R^\oplus}$$

Note that two markings are related by R^\oplus only if they have the same size; in fact, the axiom states that the empty marking is related to itself, while the rule, assuming by induction that m_1 and m_2 have the same size, ensures that $s_1 \oplus m_1$ and $s_2 \oplus m_2$ have the same size.

► **Proposition 18.** *For each relation $R \subseteq S \times S$, if $(m_1, m_2) \in R^\oplus$, then $|m_1| = |m_2|$.*

Note also that the membership $(m_1, m_2) \in R^\oplus$ may be proved in several different ways, depending on the chosen order of the elements of the two markings and on the definition of R . For instance, if $R = \{(s_1, s_3), (s_1, s_4), (s_2, s_3), (s_2, s_4)\}$, then $(s_1 \oplus s_2, s_3 \oplus s_4) \in R^\oplus$ can be proved by means of the pairs (s_1, s_3) and (s_2, s_4) , as well as by means of (s_1, s_4) , (s_2, s_3) . An alternative way to define that two markings m_1 and m_2 are related by R^\oplus is to state that m_1 can be represented as $s_1 \oplus s_2 \oplus \dots \oplus s_k$, m_2 can be represented as $s'_1 \oplus s'_2 \oplus \dots \oplus s'_k$ and $(s_i, s'_i) \in R$ for $i = 1, \dots, k$.

► **Proposition 19** ([14]). *For each place relation $R \subseteq S \times S$, the following hold:*

1. *If R is an equivalence relation, then R^\oplus is an equivalence relation.*
2. *If $R_1 \subseteq R_2$, then $R_1^\oplus \subseteq R_2^\oplus$, i.e., the additive closure is monotone.*
3. *If $(m_1, m_2) \in R^\oplus$ and $(m'_1, m'_2) \in R^\oplus$, then $(m_1 \oplus m'_1, m_2 \oplus m'_2) \in R^\oplus$, i.e., the additive closure is additive.*
4. *If R is an equivalence relation and, moreover, $(m_1 \oplus m'_1, m_2 \oplus m'_2) \in R^\oplus$ and $(m_1, m_2) \in R^\oplus$, then $(m'_1, m'_2) \in R^\oplus$, i.e., the additive closure of an equivalence place relation is subtractive.*

When R is an equivalence relation, it is rather easy to check whether two markings are related by R^\oplus . An algorithm, described in [14], establishes whether an R -preserving bijection exists between the two markings m_1 and m_2 of equal size k in $O(k^2)$ time. Another algorithm, described in [22, 15], checks whether $(m_1, m_2) \in R^\oplus$ in $O(n)$ time, where n is the size of S . However, these performant algorithms heavily rely on the fact that R is an equivalence relation, hence also subtractive (case 4 of Proposition 19). If R is not an equivalence relation, which is typical for place bisimulations, the naive algorithm for checking whether $(m_1, m_2) \in R^\oplus$ would simply consider m_1 represented as $s_1 \oplus s_2 \oplus \dots \oplus s_k$, and then would scan all the possible permutations of m_2 , each represented as $s'_1 \oplus s'_2 \oplus \dots \oplus s'_k$, to check that $(s_i, s'_i) \in R$ for $i = 1, \dots, k$. Of course, this naive algorithm has worst-case complexity $O(k!)$.

► **Example 20.** Consider $R = \{(s_1, s_3), (s_1, s_4), (s_2, s_4)\}$, which is not an equivalence relation. Suppose we want to check that $(s_1 \oplus s_2, s_4 \oplus s_3) \in R^\oplus$. If we start by matching $(s_1, s_4) \in R$, then we fail because the residual (s_2, s_3) is not in R . However, if we permute the second marking to $s_3 \oplus s_4$, then we succeed because the required pairs (s_1, s_3) and (s_2, s_4) are both in R .

Nonetheless, the problem of checking whether $(m_1, m_2) \in R^\oplus$ has polynomial time complexity because it can be considered as an instance of the problem of finding a perfect matching in a bipartite graph, where the nodes of the two partitions are the tokens in the two markings, and the edges are defined by the relation R . In fact, the definition of the bipartite graph takes $O(k^2)$ time (where $k = |m_1| = |m_2|$) and, then, the Hopcroft-Karp-Karzanov algorithm [18] for computing the maximum matching has worst-case time complexity $O(h\sqrt{k})$, where h is the number of the edges in the bipartite graph ($h \leq k^2$) and to check whether the maximum matching is perfect can be done simply by checking that the size of the matching equals the number of nodes in each partition, i.e., k . Hence, in evaluating the complexity of the algorithm in Section 4, we assume that the complexity of checking whether $(m_1, m_2) \in R^\oplus$ is in $O(k^2\sqrt{k})$.

A related problem is that of computing, given a marking m_1 of size k , the set of all the markings m_2 such that $(m_1, m_2) \in R^\oplus$. This problem can be solved with a worst-case time complexity of $O(n^k)$ because each of the k tokens in m_1 can be related via R to n places at most.

Now we list some necessary, and less obvious, properties of additively closed place relations that will be useful in the following.

► **Proposition 21** ([14]). *For each family of place relations $R_i \subseteq S \times S$, the following hold:*

1. $\emptyset^\oplus = \{(\emptyset, \emptyset)\}$, i.e., the additive closure of the empty place relation yields a singleton marking relation, relating the empty marking to itself.
2. $(\mathcal{I}_S)^\oplus = \mathcal{I}_M$, i.e., the additive closure of the identity on places $\mathcal{I}_S = \{(s, s) \mid s \in S\}$ yields the identity relation on markings $\mathcal{I}_M = \{(m, m) \mid m \in \mathcal{M}(S)\}$.
3. $(R^\oplus)^{-1} = (R^{-1})^\oplus$, i.e., the inverse of an additively closed relation R equals the additive closure of its inverse R^{-1} .
4. $(R_1 \circ R_2)^\oplus = (R_1^\oplus) \circ (R_2^\oplus)$, i.e., the additive closure of the composition of two place relations equals the compositions of their additive closures.

3.2 Pti-place bisimulation and its properties

We are now ready to introduce pti-place bisimulation, which is a non-interleaving behavioral relation defined over the net places. Note that for P/T nets, place bisimulation [4, 16] and pti-place bisimulation coincide because $I = \emptyset$.

► **Definition 22** (Pti-place bisimulation). *Let $N = (S, A, T, I)$ be a PTI net. A pti-place bisimulation is a relation $R \subseteq S \times S$ such that if $(m_1, m_2) \in R^\oplus$ then*

1. $\forall t_1$ such that $m_1[t_1]m'_1$, $\exists t_2$ such that $m_2[t_2]m'_2$ and
 - (a) $(\bullet t_1, \bullet t_2) \in R^\oplus$, $(t_1^\bullet, t_2^\bullet) \in R^\oplus$, $l(t_1) = l(t_2)$, and $(m_1 \ominus \bullet t_1, m_1 \ominus \bullet t_2) \in R^\oplus$,
 - (b) $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$.
2. $\forall t_2$ such that $m_2[t_2]m'_2$, $\exists t_1$ such that $m_1[t_1]m'_1$ and
 - (a) $(\bullet t_1, \bullet t_2) \in R^\oplus$, $(t_1^\bullet, t_2^\bullet) \in R^\oplus$, $l(t_1) = l(t_2)$, and $(m_1 \ominus \bullet t_1, m_1 \ominus \bullet t_2) \in R^\oplus$,
 - (b) $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$.

Two markings m_1 and m_2 are pti-place bisimilar, denoted by $m_1 \sim_p m_2$, if there exists a pti-place bisimulation R such that $(m_1, m_2) \in R^\oplus$.

Note that, by additivity of R^\oplus (cf. Proposition 19), from $(m_1 \ominus \bullet t_1, m_2 \ominus \bullet t_2) \in R^\oplus$ and $(t_1^\bullet, t_2^\bullet) \in R^\oplus$ we derive $(m_1', m_2') \in R^\oplus$, which is the condition required in the original definition of place bisimulation in [4]. Our slightly stronger formulation is more adequate for the proof of Theorem 27.

Conditions 1(b) and 2(b) make sure that the relation R respects the inhibiting behavior of places. Indeed, an inhibiting place for one of the two transitions cannot be related via R to a non-inhibiting place for the other transition. These conditions might appear rather restrictive, and one may wonder whether they can be weakened or omitted altogether. However, their presence is strictly necessary in the crucial step of the proof of Lemma 29. Moreover, these conditions are also essential for proving Theorem 27.

► **Example 23.** Consider the PTI net N_1 in Figure 2. Not only the loop labeled by b on the left is unwound on the right, but also the a -labeled transition on the left is replicated three times on the right. The relation

$$R = \{(s_0, s_5), (s_1, s_{11}), (s_2, s_4), (s_2, s_7), (s_3, s_6), (s_3, s_8), (s_3, s_9), (s_3, s_{10})\}$$

is a pti-place bisimulation and so, e.g., $2 \cdot s_2 \oplus s_3 \sim_p s_4 \oplus s_7 \oplus s_9$.

Now consider the PTI net N_2 in Figure 2. In this case, the b -labeled transition on the left can be matched by the b -labeled transition on the right, even if their inhibiting set differ in size, because both (s_1, s_1') and (s_3, s_1') are in the following bisimulation. Indeed, the relation

$$R' = \{(s_1, s_1'), (s_2, s_2'), (s_3, s_1'), (s_4, s_4'), (s_5, s_4'), (s_6, s_4')\}$$

is a pti-place bisimulation and so, e.g., $s_1 \oplus s_3 \oplus 2 \cdot s_2 \oplus s_5 \sim_p 2 \cdot s_1' \oplus 2 \cdot s_2' \oplus s_4'$.

► **Example 24.** Consider the PTI net in Figure 3, depicting two models of unbounded producer-consumer with priority.

In the left part, denoted as *UPAC* for readability, the producer p can generate two products of type a and b and stock them in w_a and w_b (for “warehouse”) respectively. Transitions o_a and o_b model the order of a client c from the warehouse, which may then be shipped (place s) and delivered (transition d). Product a has priority both in the production and ordering phases, and this is modelled by two inhibiting arcs between w_a and b and o_b . Roughly speaking, if there is an a in the warehouse, then no b can be produced or ordered. Moreover, only one product of type a can be stored in either w_a or w_a' , as the inhibiting arcs between a warehouse for a and the other a -transition do not allow to perform the latter until the former has been freed by the execution of transition o_a .

In the right part, denoted as *UPBC*, we duplicate the production and ordering phases of product b , and remove one of the two lines of product a . The behavior of the system remains the same, and this is proved by the pti-place bisimulation

$$R = \{(p, \bar{p}), (w_a, \bar{w}_a), (w_a', \bar{w}_a), (w_b, \bar{w}_b), (w_b, \bar{w}_b'), (s, \bar{s}), (c, \bar{c})\}.$$

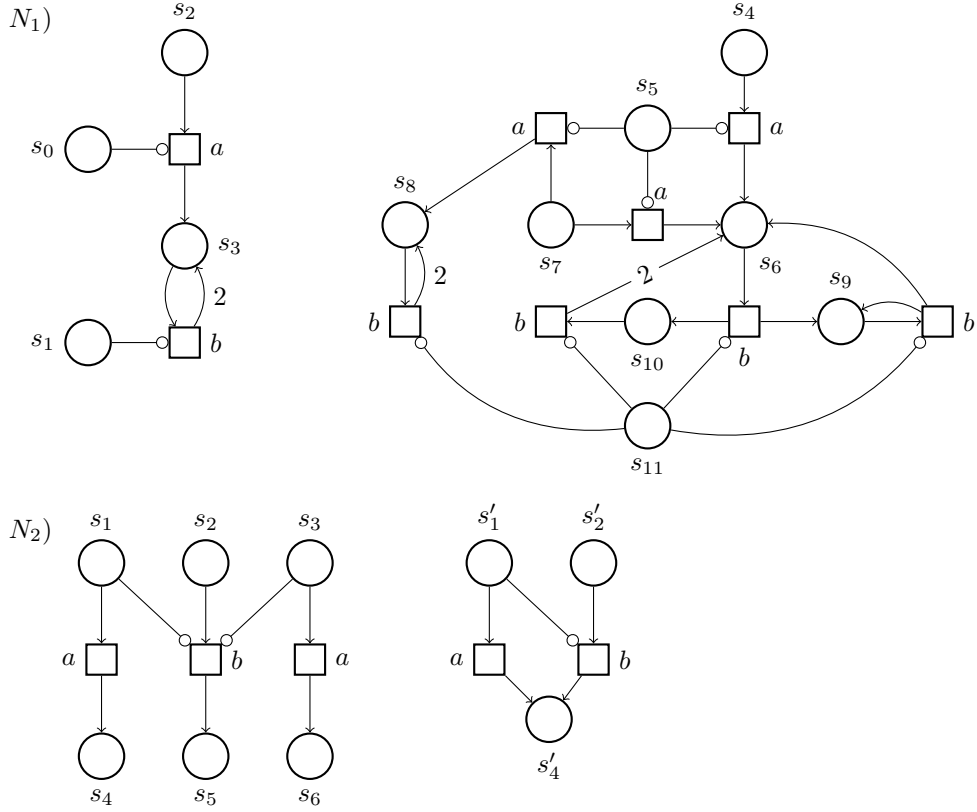
We now prove that \sim_p is an equivalence relation.

► **Proposition 25.** *For each PTI net $N = (S, A, T, I)$, relation $\sim_p \subseteq \mathcal{M}(S) \times \mathcal{M}(S)$ is an equivalence relation.*

Proof. Direct consequence of the fact that for each PTI net $N = (S, A, T, I)$, the following hold:

1. The identity relation $\mathcal{I}_S = \{(s, s) \mid s \in S\}$ is a pti-place bisimulation;
2. the inverse relation $R^{-1} = \{(s', s) \mid (s, s') \in R\}$ of a pti-place bisimulation R is a pti-place bisimulation;

28:10 A Decidable Equivalence



■ **Figure 2** Two PTI nets, whose transitions are labeled either by a or by b .

3. the relational composition $R_1 \circ R_2 = \{(s, s'') \mid \exists s'. (s, s') \in R_1 \wedge (s', s'') \in R_2\}$ of two pti-place bisimulations R_1 and R_2 is a pti-place bisimulation.

See Appendix A.1 for details. ◀

By Definition 22, pti-place bisimilarity can be defined in the following way:

$$\sim_p = \bigcup \{R^\oplus \mid R \text{ is a pti-place bisimulation}\}.$$

By monotonicity of the additive closure (Proposition 19(2)), if $R_1 \subseteq R_2$, then $R_1^\oplus \subseteq R_2^\oplus$.

Hence, we can restrict our attention to maximal pti-place bisimulations only:

$$\sim_p = \bigcup \{R^\oplus \mid R \text{ is a maximal pti-place bisimulation}\}.$$

However, it is not true that

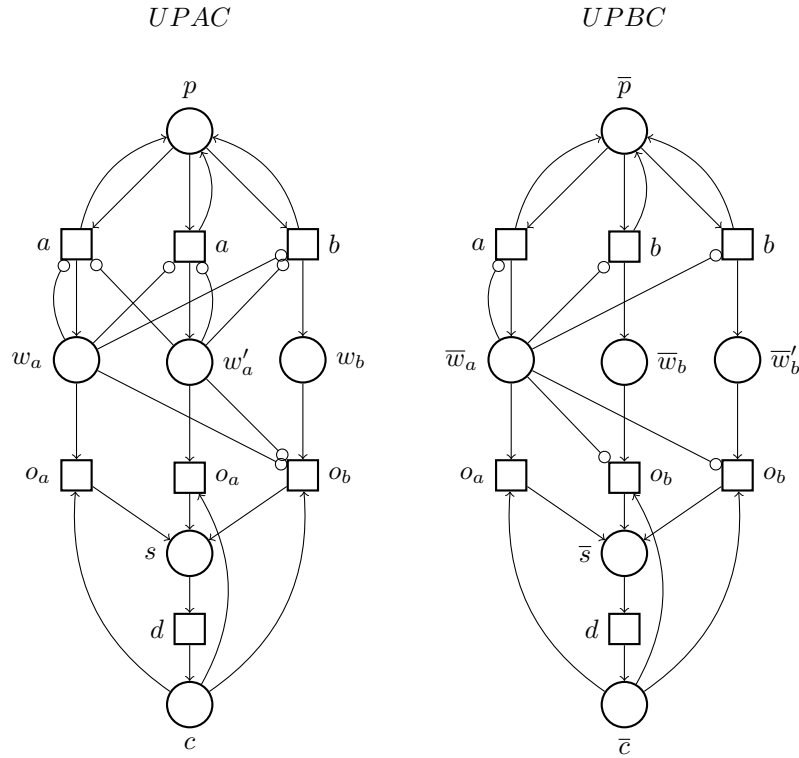
$$\sim_p = \left(\bigcup \{R \mid R \text{ is a maximal pti-place bisimulation}\} \right)^\oplus$$

because the union of pti-place bisimulations may be not a pti-place bisimulation (as already observed for place bisimulation in [4, 16]), so that its definition is not coinductive.

► **Example 26.** Consider the net in Figure 4, whose transitions are $t_1 = (s_2, s_3) \xrightarrow{a} s_1$, $t_2 = (s_2 \oplus s_3, \theta) \xrightarrow{a} s_5$ and $t_3 = (s_3, s_2) \xrightarrow{a} s_4$. Clearly, R_1 and R_2 , defined as follows, are both maximal pti-place bisimulations.

$$R_1 = \{(s_2, s_2), (s_3, s_3)\} \cup (\{s_1, s_4, s_5\} \times \{s_1, s_4, s_5\})$$

$$R_2 = \{(s_2, s_3), (s_3, s_2)\} \cup (\{s_1, s_4, s_5\} \times \{s_1, s_4, s_5\})$$



■ **Figure 3** A PTI net representing two unbounded producers/consumers with priority. For simplicity, we display the labels of transitions instead of their names.

Note that the union $R = R_1 \cup R_2$ is not a pti-place bisimulation as, for example, $(2 \cdot s_2, s_2 \oplus s_3) \in R^\oplus$, but the pti-place bisimulation conditions are not satisfied. Indeed, if $2 \cdot s_2$ moves first by $2 \cdot s_2[t_1]s_1 \oplus s_2$, then $s_2 \oplus s_3$ can only try to respond with $s_2 \oplus s_3[t_2]s_5$ since t_1 and t_3 are inhibited. However, this is not possible because we have that $(\bullet t_1, \bullet t_2) \notin R^\oplus$, and, even worse, $(s_2, \theta) \notin R^\oplus$.

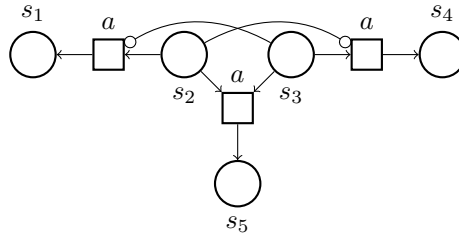
3.3 Pti-place bisimilarity is finer than causal-net bisimilarity

► **Theorem 27** (Pti-place bisimilarity implies causal-net bisimilarity). *Let $N = (S, A, T, I)$ be a PTI net and m_1, m_2 two of its markings. If $m_1 \sim_p m_2$, then $m_1 \sim_{cn} m_2$.*

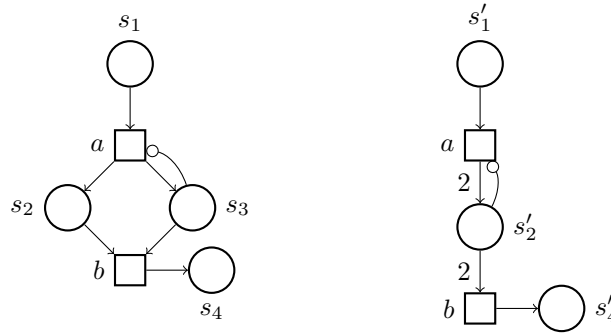
Proof. See Appendix A.2. ◀

There are at least the following three important technical differences between causal-net bisimilarity and pti-place bisimilarity.

1. A causal-net bisimulation is a very complex relation – composed of cumbersome triples of the form (ρ_1, C, ρ_2) – that must contain infinitely many triples if the net system offers a never-ending behavior. On the contrary, a pti-place bisimulation is always a very simple finite relation over the finite set S of places.
2. A causal net bisimulation proving that $m_1 \sim_{cn} m_2$ is a relation specifically designed for showing that m_1 and m_2 generate the same causal nets, step by step. If we want to prove that, e.g., $n \cdot m_1$ and $n \cdot m_2$ are causal-net bisimilar (which may not hold!), we have to construct a new causal-net bisimulation to this aim. Instead, a pti-place bisimulation



■ Figure 4 A PTI net.



■ Figure 5 Two PTI nets.

R relates those places which are considered equivalent under all the possible R -related contexts. Hence, if R justifies that $m_1 \sim_p m_2$ as $(m_1, m_2) \in R^\oplus$, then for sure the same R justifies that $n \cdot m_1$ and $n \cdot m_2$ are pti-place bisimilar, as also $(n \cdot m_1, n \cdot m_2) \in R^\oplus$.

3. Finally, while pti-place bisimilarity is decidable (see the next section), it is not known whether causal-net bisimilarity is decidable on finite PTI nets.¹

However, these technical advantages of pti-place bisimilarity over causal-net bisimilarity are balanced by an increased discriminating power of the former over the latter, that, in some cases, might appear even excessive, as the following intriguing example shows.

► **Example 28.** Consider the net in Figure 5. First of all, note that $s_2 \sim_{cn} s'_2$, because both are stuck markings. However, we have that $2 \cdot s_2 \not\sim_{cn} 2 \cdot s'_2$ because $2 \cdot s_2$ is stuck, while $2 \cdot s'_2$ can perform b . This observation is enough to conclude that $s_2 \not\sim_p s'_2$, because a pti-place bisimulation R relates places that are equivalent under any R -related context: if $(s_2, s'_2) \in R$ then $(2 \cdot s_2, 2 \cdot s'_2) \in R^\oplus$, but these two markings do not satisfy the pti-place bisimulation conditions, so R is not a pti-place bisimulation.

Nonetheless, it is interesting to observe that $s_1 \sim_{cn} s'_1$, because they generate the same causal PTI nets, step by step; moreover, even for any $n \geq 1$ we have $n \cdot s_1 \sim_{cn} n \cdot s'_1$. However, $s_1 \not\sim_p s'_1$ because it is not possible to build a pti-place bisimulation R containing the pair (s_1, s'_1) . The problem is that it would be necessary to include, into the candidate pti-place relation R , also the pair (s_2, s'_2) , which is not a pti-place bisimulation pair, as discussed above. Therefore, no pti-place bisimulation R can relate s_1 and s'_1 .

¹ Esparza observed [12] that, for finite P/T nets with at least two unbounded places, all the behavioral relations ranging from interleaving bisimilarity to fully-concurrent bisimilarity [6] are undecidable. Even if his proof does not apply to causal-net bisimilarity, we conjecture that this equivalence is undecidable as well.

4 Pti-place bisimilarity is decidable

In order to prove that \sim_p is decidable, we first need a technical lemma which states that it is decidable to check whether a place relation $R \subseteq S \times S$ is a pti-place bisimulation.

► **Lemma 29.** *Given a finite PTI net $N = (S, A, T, I)$ and a place relation $R \subseteq S \times S$, it is decidable whether R is a pti-place bisimulation.*

Proof. It is enough to check two finite conditions on transitions and places of the net; full detail in Appendix A.3. ◀

► **Theorem 30** (Pti-place bisimilarity is decidable). *Given a PTI net $N = (S, A, T, I)$, for each pair of markings m_1 and m_2 , it is decidable whether $m_1 \sim_p m_2$.*

Proof. If $|m_1| \neq |m_2|$, then $m_1 \not\sim_p m_2$ by Proposition 18. Otherwise, we can assume that $|m_1| = k = |m_2|$. As $|S| = n$, the set of all the place relations over S is of size 2^n . Let us list such relations as: R_1, R_2, \dots, R_{2^n} . Hence, for $i = 1, \dots, 2^n$, by Lemma 29 we can decide whether the place relation R_i is a pti-place bisimulation and, in such a case, we can check whether $(m_1, m_2) \in R_i^\oplus$ in $O(k^2\sqrt{k})$ time. As soon as we have found a pti-place bisimulation R_i such that $(m_1, m_2) \in R_i^\oplus$, we stop concluding that $m_1 \sim_p m_2$. If none of the R_i is a pti-place bisimulation such that $(m_1, m_2) \in R_i^\oplus$, then we can conclude that $m_1 \not\sim_p m_2$. Since this procedure might scan all place relations, the worst-case complexity of the algorithm is exponential in the number of places n . ◀

5 Conclusion

Pti-place bisimilarity is the only decidable behavioral equivalence for finite PTI nets, which constitute a powerful, Turing-complete distributed model of computation, widely used in theory and applications of concurrency (e.g., [1, 3, 9, 7, 8, 17, 19, 26]). Thus, it is the only equivalence for which it is possible (at least, in principle) to verify algorithmically the (causality-preserving) correctness of an implementation by exhibiting a pti-place bisimulation between its specification and implementation. It is also sensible, because it respects the causal behavior of PTI nets, since it is finer than causal-net bisimilarity. Of course, pti-place bisimilarity is a rather discriminating behavioral equivalence, as illustrated in Example 28, and a proper evaluation of its usefulness on real case studies is left for future research.

In our interpretation, (pti-)place bisimilarity is an attempt of giving semantics to *unmarked*, rather than marked, nets, shifting the focus from the usually undecidable question *When are two markings equivalent?* to the decidable (but more restrictive) question *When are two places equivalent?* A possible answer to the latter question may be: two places are equivalent if, whenever the same number of tokens are put on these two places, the behavior of the marked nets is the same. If we reinterpret Example 28 in this perspective, we clearly see that place s_2 and place s'_2 cannot be considered as equivalent because, even if the marking s_2 and s'_2 are equivalent (as they are both stuck), the marking $2 \cdot s_2$ is not equivalent to the marking $2 \cdot s'_2$ (as only the latter can move). More specifically, a (pti-)place bisimulation R considers two places s_1 and s_2 as equivalent if $(s_1, s_2) \in R$, as, by definition of (pti-)place bisimulation, they must behave the same in any R -related context.

The decidability result for pti-place bisimilarity is based on the fact that the net model is finite, even if the associated reachability graph may be unboundedly large or even infinite: indeed, one can decide pti-place bisimilarity simply checking a large, but finite, number of conditions on the shape of the finite net, rather than inspecting its (possibly, infinitely many) reachable markings.

Turing completeness is achieved in PTI nets by means of their ability to test for zero. Other Turing-complete models of computation may exploit different mechanisms to this aim. For instance, in the π -calculus [24, 28] Turing completeness is achieved by means of the ability to generate unboundedly new names (by means of the interplay between recursion and the restriction operator), but this feature is not describable by means of a finite net model [8, 23]. For this reason, we think it is hard to find a sensible, decidable behavioral equivalence for the whole π -calculus.

To the best of our knowledge, this is the second paper proving the decidability of a behavioral equivalence for a Turing-complete formalism. In fact, in [21] it is proved that (interleaving) bisimilarity is decidable for a small process calculus, called HOcore, with higher-order communication (but without restriction), that is, nonetheless, Turing-complete.

Future work will be devoted to see whether the pti-place bisimulation idea can be extended to other, possibly even larger classes of nets, such as *lending* Petri nets [5], where transitions are allowed to consume tokens from a place even if it does not contain enough tokens, thus enabling negative-valued markings.

References

- 1 Tilak Agerwala. A complete model for representing the coordination of asynchronous processes. *Technical report*, 1974. doi:10.2172/4242290.
- 2 Tilak Agerwala and Mike Flynn. Comments on capabilities, limitations and “correctness” of petri nets. *SIGARCH Computer Architecture News*, 2(4):81–86, 1973. doi:10.1145/633642.803973.
- 3 Marco Ajmone, Gianfranco Balbo, Gianni Conte, Susanna Donatelli, and Giuliana Franceschinis. Modelling with generalized stochastic petri nets. *SIGMETRICS Performance Evaluation Review*, 26(2):2, 1998. doi:10.1145/288197.581193.
- 4 Cyril Autant, Z. Belmesk, and Philippe Schnoebelen. Strong bisimilarity on nets revisited. In *PARLE '91 Parallel Architectures and Languages Europe*, volume 506 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 1991. doi:10.1007/3-540-54152-7_71.
- 5 Massimo Bartoletti, Tiziana Cimoli, and G. Michele Pinna. Lending petri nets. *Science of Computer Programming*, 112:75–101, 2015. doi:10.1016/j.scico.2015.05.006.
- 6 Eike Best, Raymond R. Devillers, Astrid Kiehn, and Lucia Pomello. Concurrent bisimulations in petri nets. *Acta Informatica*, 28(3):231–264, 1991. doi:10.1007/BF01178506.
- 7 Nadia Busi. Analysis issues in petri nets with inhibitor arcs. *Theoretical Computer Science*, 275(1):127–177, 2002. doi:10.1016/S0304-3975(01)00127-X.
- 8 Nadia Busi and Roberto Gorrieri. Distributed semantics for the π -calculus based on petri nets with inhibitor arcs. *The Journal of Logic and Algebraic Programming*, 78(3):138–162, 2009. doi:10.1016/j.jlap.2008.08.002.
- 9 Nadia Busi and G. Michele Pinna. Process semantics for place/transition nets with inhibitor and read arcs. *Fundamenta Informaticae*, 40:165–197, 1999. doi:10.3233/FI-1999-402304.
- 10 Nadia Busi and G. Michele Pinna. Comparing truly concurrent semantics for contextual place/transition nets. *Fundamenta Informaticae*, 44(3):209–244, 2000. doi:10.5555/2376335.2376336.
- 11 Arnaldo Cesco and Roberto Gorrieri. A decidable equivalence for a turing-complete, distributed model of computation, 2021. arXiv:2104.14859.
- 12 Javier Esparza. *Decidability and complexity of Petri net problems – An introduction*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer Berlin Heidelberg, 1998. doi:10.1007/3-540-65306-6_20.
- 13 Ursula Goltz and Wolfgang Reisig. The non-sequential behaviour of petri nets. *Information and Control*, 57(2):125–147, 1983. doi:10.1016/S0019-9958(83)80040-0.

- 14 Roberto Gorrieri. Team bisimilarity, and its associated modal logic, for bpp nets. *Acta Informatica*, 2020. doi:10.1007/s00236-020-00377-4.
- 15 Roberto Gorrieri. Causal semantics for BPP nets with silent moves. *Fundamenta Informaticae*, 180(3):179–249, 2021. doi:10.3233/FI-2021-2039.
- 16 Roberto Gorrieri. Place bisimilarity is decidable, indeed!, 2021. arXiv:2104.01392.
- 17 M. Hack. Petri net languages. Technical report, MIT, 1976. doi:10.5555/888947.
- 18 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973. doi:10.1137/0202019.
- 19 Ryszard Janicki and Maciej Koutny. Semantics of inhibitor nets. *Information and Computation*, 123(1):1–16, 1995. doi:10.1006/inco.1995.1153.
- 20 Petr Jančár. Undecidability of bisimilarity for petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995. doi:10.1016/0304-3975(95)00037-W.
- 21 Ivan Lanese, Jorge A. Pérez, Davide Sangiorgi, and Alan Schmitt. On the expressiveness and decidability of higher-order process calculi. *Information and Computation*, 209(2):198–226, 2011. doi:10.1016/j.ic.2010.10.001.
- 22 Alessandro Liberato. A study on bisimulation equivalence and team equivalence. Master thesis (supervisor R. Gorrieri), 2019.
- 23 Roland Meyer and Roberto Gorrieri. On the relationship between π -calculus and finite place/transition petri nets. In *CONCUR 2009 – Concurrency Theory*, volume 5710 of *Lecture Notes in Computer Science*, pages 463–480. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-04081-8_31.
- 24 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes. *Information and Computation*, 100(1):1–77, 1992. doi:10.1016/0890-5401(92)90008-4.
- 25 Ernst R. Olderog. *Nets, Terms and Formulas*, volume Cambridge Tracts in Theoretical Computer Science 23. Cambridge University Press, 1991. doi:10.1017/CB09780511526589.
- 26 James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1981. doi:10.5555/539513.
- 27 Wolfgang Reisig. *Petri Nets: An Introduction*. Springer-Verlag, 1985. doi:10.1007/978-3-642-69968-9.
- 28 Davide Sangiorgi and David Walker. *The π -calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.
- 29 Rob J. van Glabbeek. *Structure Preserving Bisimilarity, Supporting an Operational Petri Net Semantics of CCSP*, volume 9360 of *Lecture Notes in Computer Science*, pages 99–130. Springer International Publishing, 2015. doi:10.1007/978-3-319-23506-6_9.

A Properties of pti-place bisimilarity

A.1 Pti-place bisimilarity is an equivalence

► **Proposition 31.** *For each PTI net $N = (S, A, T, I)$, the following hold:*

1. *The identity relation $\mathcal{I}_S = \{(s, s) \mid s \in S\}$ is a pti-place bisimulation;*
2. *the inverse relation $R^{-1} = \{(s', s) \mid (s, s') \in R\}$ of a pti-place bisimulation R is a pti-place bisimulation;*
3. *the relational composition $R_1 \circ R_2 = \{(s, s'') \mid \exists s'. (s, s') \in R_1 \wedge (s', s'') \in R_2\}$ of two pti-place bisimulations R_1 and R_2 is a pti-place bisimulation.*

Proof. The proof is almost standard, due to Proposition 21.

(1) \mathcal{I}_S is a pti-place bisimulation as for each $(m, m) \in \mathcal{I}_S^\oplus$ whatever transition t the left (or right) marking m performs a transition (say, $m[t]m'$), the right (or left) instance of m in the pair does exactly the same transition $m[t]m'$ and, of course, $(\bullet t, \bullet t) \in \mathcal{I}_S^\oplus$, $(t \bullet, t \bullet) \in \mathcal{I}_S^\oplus$, $l(t) = l(t)$, $(m \ominus \bullet t, m \ominus \bullet t) \in \mathcal{I}_S^\oplus$, by Proposition 21(2), and, also, $\forall s \in S. (s, s) \in \mathcal{I}_S \Rightarrow (s \in \circ t \Leftrightarrow s \in \circ t)$, as required by the pti-place bisimulation definition.

(2) Suppose $(m_2, m_1) \in (R^{-1})^\oplus$ and $m_2[t_2]m'_2$. By Proposition 21(3) $(m_2, m_1) \in (R^\oplus)^{-1}$ and so $(m_1, m_2) \in R^\oplus$. Since R is a pti-place bisimulation, item 2 of the bisimulation game ensures that there exist t_1 and m'_1 such that $m_1[t_1]m'_1$, with $(\bullet t_1, \bullet t_2) \in R^\oplus$, $l(t_1) = l(t_2)$, $(t_1^\bullet, t_2^\bullet) \in R^\oplus$ and $(m_1 \ominus \bullet t_1, m_2 \ominus \bullet t_2) \in R^\oplus$; moreover, $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in {}^\circ t_1 \Leftrightarrow s' \in {}^\circ t_2)$. Summing up, if $(m_2, m_1) \in (R^{-1})^\oplus$, to the move $m_2[t_2]m'_2$, m_1 replies with the move $m_1[t_1]m'_1$, such that (by Proposition 21(3)) $(\bullet t_2, \bullet t_1) \in (R^{-1})^\oplus$, $l(t_2) = l(t_1)$, $(t_2^\bullet, t_1^\bullet) \in (R^{-1})^\oplus$, $(m_2 \ominus \bullet t_2, m_1 \ominus \bullet t_1) \in (R^{-1})^\oplus$ and, moreover, $\forall s, s' \in S. (s', s) \in R^{-1} \Rightarrow (s' \in {}^\circ t_2 \Leftrightarrow s \in {}^\circ t_1)$, as required. The case when m_1 moves first is symmetric and thus omitted.

(3) Suppose $(m, m'') \in (R_1 \circ R_2)^\oplus$ and $m[t_1]m_1$. By Proposition 21(4), we have that $(m, m'') \in R_1^\oplus \circ R_2^\oplus$, and so there exists m' such that $(m, m') \in R_1^\oplus$ and $(m', m'') \in R_2^\oplus$. As $(m, m') \in R_1^\oplus$ and R_1 is a pti-place bisimulation, if $m[t_1]m_1$, then there exist t_2 and m_2 such that $m'[t_2]m_2$ with $(\bullet t_1, \bullet t_2) \in R_1^\oplus$, $l(t_1) = l(t_2)$, $(t_1^\bullet, t_2^\bullet) \in R_1^\oplus$ and $(m \ominus \bullet t_1, m' \ominus \bullet t_2) \in R_1^\oplus$; moreover, $\forall s, s' \in S. (s, s') \in R_1 \Rightarrow (s \in {}^\circ t_1 \Leftrightarrow s' \in {}^\circ t_2)$. But as $(m', m'') \in R_2^\oplus$ and R_2 is a pti-place bisimulation, we have also that there exist t_3 and m_3 such that $m''[t_3]m_3$ with $(\bullet t_2, \bullet t_3) \in R_2^\oplus$, $l(t_2) = l(t_3)$, $(t_2^\bullet, t_3^\bullet) \in R_2^\oplus$ and $(m' \ominus \bullet t_2, m'' \ominus \bullet t_3) \in R_2^\oplus$; moreover, $\forall s', s'' \in S. (s', s'') \in R_2 \Rightarrow (s' \in {}^\circ t_2 \Leftrightarrow s'' \in {}^\circ t_3)$. Summing up, for $(m, m'') \in (R_1 \circ R_2)^\oplus$, if $m[t_1]m_1$, then there exist t_3 and m_3 such that $m''[t_3]m_3$ and (by Proposition 21(4)) $(\bullet t_1, \bullet t_3) \in (R_1 \circ R_2)^\oplus$, $l(t_1) = l(t_3)$, $(t_1^\bullet, t_3^\bullet) \in (R_1 \circ R_2)^\oplus$ and $(m \ominus \bullet t_1, m'' \ominus \bullet t_3) \in (R_1 \circ R_2)^\oplus$; moreover, $\forall s, s'' \in S. (s, s'') \in R_1 \circ R_2 \Rightarrow (s \in {}^\circ t_1 \Leftrightarrow s'' \in {}^\circ t_3)$, as required. The case when m'' moves first is symmetric and so omitted. \blacktriangleleft

► **Proposition 32.** *For each PTI net $N = (S, A, T, I)$, relation $\sim_p \subseteq \mathcal{M}(S) \times \mathcal{M}(S)$ is an equivalence relation.*

Proof. Direct consequence of Proposition 31. \blacktriangleleft

A.2 Pti-place bisimilarity is finer than causal-net bisimilarity

► **Theorem 33** (Pti-place bisimilarity implies causal-net bisimilarity). *Let $N = (S, A, T, I)$ be a PTI net and m_1, m_2 two of its markings. If $m_1 \sim_p m_2$, then $m_1 \sim_{cn} m_2$.*

Proof. If $m_1 \sim_p m_2$, then there exists a pti-bisimulation R_1 such that $(m_1, m_2) \in R_1^\oplus$. Let us consider

$$R_2 \stackrel{def}{=} \{(\rho_1, C, \rho_2) \mid (C, \rho_1) \text{ is a PTI process of } N(m_1) \text{ and} \\ (C, \rho_2) \text{ is a PTI process of } N(m_2) \text{ and} \\ \forall b \in B (\rho_1(b), \rho_2(b)) \in R_1\}.$$

We want to prove that R_2 is a causal-net bisimulation. First of all, consider a triple of the form $(\rho_1^0, C^0, \rho_2^0)$, where C^0 is the causal PTI net without events and ρ_1^0, ρ_2^0 are such that $\rho_i^0(\text{Min}(C^0)) = \rho_i^0(\text{Max}(C^0)) = \rho_i^0(B^0) = m_i$ for $i = 1, 2$, and $(\rho_1^0(b), \rho_2^0(b)) \in R_1$ for all $b \in B^0$. Then $(\rho_1^0, C^0, \rho_2^0)$ must belong to R_2 , because (C^0, ρ_i^0) is a process of $N(m_i)$, for $i = 1, 2$ and, by hypothesis, $(m_1, m_2) \in R_1^\oplus$. Hence, if R_2 is a causal-net bisimulation, then the triple $(\rho_1^0, C^0, \rho_2^0) \in R_2$ ensures that $m_1 \sim_{cn} m_2$.

Assume $(\rho_1, C, \rho_2) \in R_2$. In order for R_2 to be a cn-bisimulation, we must prove that

- (i) $\forall t_1, C', \rho'_1$ such that $(C, \rho_1) \xrightarrow{e} (C', \rho'_1)$, where $\rho'_1(e) = t_1$, $\exists t_2, \rho'_2$ such that $(C, \rho_2) \xrightarrow{e} (C', \rho'_2)$, where $\rho'_2(e) = t_2$, and $(\rho'_1, C', \rho'_2) \in R_2$;
- (ii) symmetrical, if (C, ρ_2) moves first.

Assume $(C, \rho_1) \xrightarrow{e} (C', \rho'_1)$ with $\rho'_1(e) = t_1$. Since $(\rho_1, C, \rho_2) \in R_2$, for all $b \in \text{Max}(C)$ we have $(\rho_1(b), \rho_2(b)) \in R_1$ and therefore $(\rho_1(\text{Max}(C)), \rho_2(\text{Max}(C))) \in R_1^\oplus$. Since $\rho_1(\text{Max}(C)) [t_1] \rho'_1(\text{Max}(C'))$ and R_1 is a pti-place bisimulation, there exist t_2, m_2 such that $\rho_2(\text{Max}(C)) [t_2] m_2$ with $(\bullet t_1, \bullet t_2) \in R_1^\oplus$, $l(t_1) = l(t_2)$, $(t_1^\bullet, t_2^\bullet) \in R_1^\oplus$, $(\rho_1(\text{Max}(C)) \ominus \bullet \rho'_1(e), \rho_2(\text{Max}(C)) \ominus \bullet \rho'_2(e)) \in R_1^\oplus$ and, moreover, $\forall s, s' \in S. (s, s') \in R_1 \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$. Note that, since $(t_1^\bullet, t_2^\bullet) \in R_1^\oplus$ and $(\rho_1(\text{Max}(C)) \ominus \bullet \rho'_1(e), \rho_2(\text{Max}(C)) \ominus \bullet \rho'_2(e)) \in R_1^\oplus$, by additivity of additive closure (cf. Proposition 19), $(\rho_1(\text{Max}(C)) \ominus \bullet \rho'_1(e) \oplus t_1^\bullet, \rho_2(\text{Max}(C)) \ominus \bullet \rho'_2(e) \oplus t_2^\bullet) \in R_1^\oplus$, i.e. $(\rho'_1(\text{Max}(C')), m_2) \in R^\oplus$.

Therefore, since t_1 and t_2 have the same pre-sets/post-sets up to R_1 , it is possible to derive $(C, \rho_2) \xrightarrow{e} (C'', \rho'_2)$, where ρ'_2 is such that $\rho'_2(e) = t_2$ and $(\rho'_1(b), \rho'_2(b)) \in R_1$ for each $b \in e^\bullet$ (which is really possible because $(t_1^\bullet, t_2^\bullet) \in R_1^\oplus$). Now we prove that $C' = C''$. The underlying P/T parts of C' and C'' are obviously the same (so C' and C'' have the same events, the same conditions and the same flow relation), therefore we have to check that also the newly added (after/before) inhibitor arcs are the same, i.e.,

- $\forall b \in B'$ such that $b^\bullet \neq \emptyset$ we have $b \mathcal{A}_1 e \iff b \mathcal{A}_2 e$, and
- $\forall b \in e^\bullet \forall e' \in E$ we have $b \mathcal{B}_1 e' \iff b \mathcal{B}_2 e'$,

where we denote \mathcal{A}_1 (resp. \mathcal{B}_1) the after (before) inhibitor arcs obtained by extending C to C' and \mathcal{A}_2 (resp. \mathcal{B}_2) the after (before) inhibitor arcs obtained by extending C to C'' . However, these additional requests are trivially satisfied because we know that $\forall s, s' \in S. (s, s') \in R_1 \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$. In fact, if $b \mathcal{A}_1 e$, then, by Definition 13, there is an inhibitor arc from $\rho_1(b)$ to t_1 , i.e., $\rho_1(b) \in \circ t_1$. Since $(\rho_1(b), \rho_2(b)) \in R_1$, this implies that $\rho_2(b) \in \circ t_2$ and so $b \mathcal{A}_2 e$. The implication on the other side is symmetrical, and therefore omitted. The argument for relations $\mathcal{B}_1, \mathcal{B}_2$ is the same, and therefore omitted.

To conclude, we have $C' = C''$. Thus, $(C, \rho_2) \xrightarrow{e} (C', \rho'_2)$ with $\rho'_2(e) = t_2$ and $(\rho'_1(b), \rho'_2(b)) \in R_1$ for each $b \in e^\bullet$. Hence, for all $b' \in B'$ it holds that $(\rho'_1(b'), \rho'_2(b')) \in R_1$, because for all $b' \in B$ this holds by hypothesis and for all $b' \in e^\bullet$ this follows by construction (thanks to the fact that $(t_1^\bullet, t_2^\bullet) \in R_1^\oplus$). As a consequence $(\rho'_1, C', \rho'_2) \in R_2$.

The case where (C, ρ_2) moves first is symmetrical and therefore omitted. Thus, R_2 is a causal-net bisimulation and, since $(\rho_1^0, C^0, \rho_2^0) \in R_2$, we have $m_1 \sim_{cn} m_2$. ◀

A.3 It is decidable whether a place relation is a pti-place bisimulation

► **Lemma 34.** *Given a finite PTI net $N = (S, A, T, I)$ and a place relation $R \subseteq S \times S$, it is decidable whether R is a pti-place bisimulation.*

Proof. We want to prove that R is a pti-place bisimulation if and only if the following two finite conditions are satisfied:

1. $\forall t_1$ such that $\bullet t_1 [t_1]$, $\forall m$ such that $(\bullet t_1, m) \in R^\oplus$, $\exists t_2$ such that $\bullet t_2 [t_2]$ and
 - (a) $\bullet t_2 = m$,
 - (b) $(t_1^\bullet, t_2^\bullet) \in R^\oplus$, $l(t_1) = l(t_2)$,
 - (c) $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$.
2. $\forall t_2$ such that $\bullet t_2 [t_2]$, $\forall m$ such that $(m, \bullet t_2) \in R^\oplus$, $\exists t_1$ such that $\bullet t_1 [t_1]$ and
 - (a) $\bullet t_1 = m$,
 - (b) $(t_1^\bullet, t_2^\bullet) \in R^\oplus$, $l(t_1) = l(t_2)$,
 - (c) $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$.

First we prove the implication from left to right, only for condition 1, as the other is symmetrical. If R is a pti-place bisimulation and $(\bullet t_1, m) \in R^\oplus$, then from $\bullet t_1 [t_1] t_1^\bullet$ it follows that there exists t_2 such that $\bullet t_2 [t_2] t_2^\bullet$ with $\bullet t_2 = m$, $(t_1^\bullet, t_2^\bullet) \in R^\oplus$, $l(t_1) = l(t_2)$ and, moreover, $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$. Therefore, conditions (a), (b) and (c) are trivially satisfied.

Now we prove the implication from right to left, i.e., if conditions 1 and 2 hold for R , then R is a pti-place bisimulation. Suppose $(m_1, m_2) \in R^\oplus$ and $m_1[t_1]m'_1$. Let $q = \{(s_1, s'_1), (s_2, s'_2), \dots, (s_k, s'_k)\}$ be any multiset of associations that can be used to prove that $(m_1, m_2) \in R^\oplus$. So this means that $m_1 = s_1 \oplus s_2 \oplus \dots \oplus s_k$, $m_2 = s'_1 \oplus s'_2 \oplus \dots \oplus s'_k$ and that $(s_i, s'_i) \in R$ for $i = 1, \dots, k$. If $m_1[t_1]m'_1$, then $m'_1 = m_1 \ominus \bullet t_1 \oplus t_1^\bullet$. Consider the multiset of associations $p = \{(\bar{s}_1, \bar{s}'_1), \dots, (\bar{s}_h, \bar{s}'_h)\} \subseteq q$, with $\bar{s}_1 \oplus \dots \oplus \bar{s}_h = \bullet t_1$.




Note that $(\bullet t_1, \bar{s}'_1 \oplus \dots \oplus \bar{s}'_h) \in R^\oplus$ and that $\bullet t_1[t_1]$. Hence, by condition 1, there exists a transition t_2 such that $\bullet t_2[t_2]$, $\bullet t_2 = \bar{s}'_1 \oplus \dots \oplus \bar{s}'_h$, $(t_1^\bullet, t_2^\bullet) \in R^\oplus$, $l(t_1) = l(t_2)$, and $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$. By hypothesis, $\circ t_1 \cap \text{dom}(m_1) = \emptyset$, so since $(m_1, m_2) \in R^\oplus$ and condition (c) holds, we have that $\circ t_2 \cap \text{dom}(m_2) = \emptyset$. Therefore, since $\bullet t_2 \subseteq m_2$, also $m_2[t_2]m'_2$ is firable, where $m'_2 = m_2 \ominus \bullet t_2 \oplus t_2^\bullet$, and we have that $(\bullet t_1, \bullet t_2) \in R^\oplus$, $(t_1^\bullet, t_2^\bullet) \in R^\oplus$, $l(t_1) = l(t_2)$, $(m_1 \ominus \bullet t_1, m_2 \ominus \bullet t_2) \in R^\oplus$ and, moreover, $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$, as required, where $(m_1 \ominus \bullet t_1, m_2 \ominus \bullet t_2) \in R^\oplus$ holds as, from the set q of matching pairs for m_1 and m_2 , we have removed those in p .

If $m_2[t_2]m'_2$, then we have to use an argument symmetric to the above, where condition 2 is used instead. Hence, we have proved that conditions 1 and 2 are enough to prove that R is a pti-place bisimulation.

Finally, the complexity of this procedure is as follows. For condition 1, we have to consider all the net transitions, and for each t_1 we have to consider all the markings m such that $(\bullet t_1, m) \in R_i^\oplus$, and for each pair (t_1, m) we have to check whether there exists a transition t_2 such that $m = \bullet t_2$, $l(t_1) = l(t_2)$, $(t_1^\bullet, t_2^\bullet) \in R_i^\oplus$ and, moreover, that $\forall s, s' \in S. (s, s') \in R \Rightarrow (s \in \circ t_1 \Leftrightarrow s' \in \circ t_2)$. And the same for condition 2. Hence, this procedure has worst-case time complexity $O(q \cdot n^p \cdot q \cdot (p^2 \sqrt{p} + n^2 \cdot p))$, where $q = |T|$, $n = |S|$ and p is the least number such that $|\bullet t| \leq p$, $|\circ t| \leq p$, and $|t^\bullet| \leq p$ for all $t \in T$, as the number of markings m related via R_i to $\bullet t_1$ is n^p at most, checking whether $(t_1^\bullet, t_2^\bullet) \in R_i^\oplus$ takes $O(p^2 \sqrt{p})$ in the worst case and, finally, checking the conditions on the inhibiting sets is $n^2 \cdot p$ at most. ◀

Black-Box Hypotheses and Lower Bounds

Brynmor K. Chapman  
MIT, USA

R. Ryan Williams   
MIT, USA

Abstract

What sort of code is so difficult to analyze that every potential analyst can discern essentially no information from the code, other than its input-output behavior? In their seminal work on program obfuscation, Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang (CRYPTO 2001) proposed the Black-Box Hypothesis, which roughly states that every property of Boolean functions which has an efficient “analyst” and is “code independent” can also be computed by an analyst that only has black-box access to the code. In their formulation of the Black-Box Hypothesis, the “analysts” are arbitrary randomized polynomial-time algorithms, and the “codes” are general (polynomial-size) circuits. If true, the Black-Box Hypothesis would immediately imply $\text{NP} \not\subseteq \text{BPP}$.

We consider generalized forms of the Black-Box Hypothesis, where the set of “codes” \mathcal{C} and the set of “analysts” \mathcal{A} may correspond to other efficient models of computation, from more restricted models such as AC^0 to more general models such as nondeterministic circuits. We show how lower bounds of the form $\mathcal{C} \not\subseteq \mathcal{A}$ often imply a corresponding Black-Box Hypothesis for those respective codes and analysts. We investigate the possibility of “complete” problems for the Black-Box Hypothesis: problems in \mathcal{C} such that they are not in \mathcal{A} if and only if their corresponding Black-Box Hypothesis is true. Along the way, we prove an equivalence: for nondeterministic circuit classes \mathcal{C} , the “ \mathcal{C} -circuit satisfiability problem” is not in \mathcal{A} if and only if the Black-Box Hypothesis is true for analysts in \mathcal{A} .

2012 ACM Subject Classification Theory of computation \rightarrow Circuit complexity

Keywords and phrases Black-Box hypothesis, circuit complexity, lower bounds

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.29

Funding Partially supported by NSF CCF-1741615, NSF CCF-1909429, and a Frank Quick Faculty Innovation Fellowship.

1 Introduction

What kind of code “behaves” like a black box to any code analyst? In particular, what programs are so difficult to analyze that every potential analyst can discern essentially no information from the code, other than its input-output behavior? Such questions are of great importance in cryptography and formal verification: what sort of code is difficult to verify without considerable resources? What kind of code can be obfuscated? What properties of functions can be automatically tested?

A priori, the answers to such questions depend on three factors:

1. The complexity of the code: what instructions are allowed in the code, the computational complexity (e.g. time/space/size/depth complexity) of the algorithm implemented by the code, and so on.
2. The complexity of the analyst: what sorts of operations the analyst can perform, and how much resources it has (time/space/size/depth) to analyze the code.
3. The actual function being computed by the code. If the function itself is trivial or extremely complicated, this could affect how “black box” it can possibly look.



© Brynmor K. Chapman and R. Ryan Williams;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 29; pp. 29:1–29:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper, we consider these three factors carefully, and study obfuscation from a different direction compared to most existing literature on the subject. In particular, we propose generalized forms of the “Black-Box Hypothesis” considered in Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [5] and show how such questions are intimately related to lower bound questions.

A Complexity-Theoretic View

In the pioneering work of Barak et al. [5] on obfuscation, the authors also proposed a compelling conjecture about black-box obfuscation that they called a “Scaled-Down Rice’s Theorem” [5, Conjecture 5.1]; the conjecture has recently been renamed the *Black Box Hypothesis* (BBH) [20, 14]. Informally, the Black-Box Hypothesis posits that, when code is represented as a small Boolean circuit, and a code analyst is represented as an efficient (polynomial-time) randomized algorithm, the only possible analysis tasks are ones that could have been performed using only the input-output behavior of the code (and not the code itself).

While the original Black-Box Hypothesis is still a major open problem, other natural variants of the hypothesis may be tractable for us to resolve, **unconditionally**. We consider variants of the Black-Box Hypothesis in a more general complexity-theoretic setting, where the complexity of the analyst, the complexity of the code being analyzed, and the function to be obfuscated (the “box”) are carefully taken into account. For example, we consider the case where the “analyst function” is taken from a “low” complexity class \mathcal{A} (smaller than P, polynomial time), and the box is also from a “low” complexity class \mathcal{C} .

More formally, we study abstract forms of the Black-Box Hypothesis (sometimes abbreviated as BBH in the following). Let \mathcal{C} be a set of circuits and let \mathcal{A} be a complexity class that permits oracles in its definition. We say that a property $P : \mathcal{C} \rightarrow \{0, 1\}$ of \mathcal{C} is *semantic* if $P(C) = P(C')$ for all pairs of circuits C and C' in \mathcal{C} which compute the same function.

► **Hypothesis 1** (*C-Black-Box Hypothesis for \mathcal{A}*). [Informal Statement, cf. Hypothesis 14] *Let $P : \mathcal{C} \rightarrow \{0, 1\}$ be any semantic property computable by some analyst $A' \in \mathcal{A}$. Then there is a **black-box** analyst $A \in \mathcal{A}$ such that for every s and every circuit $C \in \mathcal{C}$ of size s on n inputs, $A^C(1^n 0^{s-n}) = P(C)$.*

In prior work, the class of analysts \mathcal{A} was always set to be BPP, and the class of circuits \mathcal{C} was generally set to be unrestricted circuits of fan-in two. In that full form, proving the BBH would also prove $\text{NP} \not\subseteq \text{BPP}$, so that is presently out of reach! (The BBH could also end up being false, of course.) By considering a range of natural possible choices for the weak analysts \mathcal{A} and the circuit sets \mathcal{C} , we can try to delineate precisely how weak the analysts from \mathcal{A} need to be, in order for \mathcal{C} -circuits to *provably* behave like black boxes, and to relate the corresponding Black-Box Hypotheses to other core problems within complexity.

1.1 Our Results

We demonstrate several interesting relationships between circuit lower bounds and Black-Box Hypotheses in the generalized setting. First, we prove that certain instances of the Black-Box Hypothesis are true, from known circuit lower bounds. In fact we give a generic connection from lower bounds to Black-Box Hypotheses. We also give some converse results, showing that Black-Box Hypotheses imply certain circuit lower bounds. Finally, in some settings, we can show that certain problems are “complete” for a Black-Box Hypothesis, in the sense that proving the Black-Box Hypothesis is *equivalent* to proving a lower bound against the aforementioned problem.

Black-Box Hypotheses For Restricted Analysts, From Lower Bounds

In Section 4, we explore situations in which known lower bounds imply Black-Box Hypotheses. We first consider Hypothesis 14 where the classes of analysts \mathcal{A} are restricted, and the set of potential “boxes” \mathcal{C} consists of *unrestricted circuits*. We show that one can prove a \mathcal{C} -Black-Box Hypothesis for \mathcal{A} , when the given set of boxes \mathcal{C} is sufficiently powerful and the set of analysts \mathcal{A} is limited. *We find this to be counterintuitive.* It could have been the case that, when the set of boxes \mathcal{C} is powerful, an analyst with access to the code of such a powerful box might be able to learn something interesting about it, and gain more power than if it only had black-box access. However, it turns out that when the boxes are sufficiently powerful, no analyst can learn any *semantic* property.

We find that, under very general conditions, circuit lower bounds against \mathcal{A} (as an algorithmic class) imply the Black-Box Hypothesis for \mathcal{A} (as an analyst class).

► **Theorem 2** (Informal Statement, cf. Theorem 16). *Let \mathcal{A} be a circuit class (of analysts), and let f be a Boolean function computable with (general) circuits of size at most $t(n)$. Suppose $f \notin \mathcal{A}$, and suppose \mathcal{A} is closed under projections from n variables onto $O(t(n) \log t(n))$ variables. Then the (general) Black-Box Hypothesis for \mathcal{A} is true.*

The full formal version of the theorem appears in Section 4 as Theorem 16. Intuitively, we apply a “input-switching” trick which reduces the task of computing f on an input \mathbf{y} to the task of deciding *any* non-trivial semantic property P on a circuit $D_{\mathbf{y}}$.¹ In particular, given an analyst A computing P , we show how to map every Boolean string \mathbf{y} (a potential input for f) into a circuit $D_{\mathbf{y}}$ whose input-output behavior (and in particular, whether $D_{\mathbf{y}}$ satisfies the property P) depends on the value $f(\mathbf{y})$. At a high level, $D_{\mathbf{y}}$ takes an input \mathbf{x} , evaluates $f(\mathbf{y})$, and then (depending on $f(\mathbf{y})$) evaluates and outputs either $C_1(\mathbf{x})$ or $C_2(\mathbf{x})$, where C_1 and C_2 are fixed circuits (independent of \mathbf{y}), exactly one of which satisfies the property P . In essence, we are “switching” the input \mathbf{y} with a circuit $D_{\mathbf{y}}$ which can evaluate f , and for which we can determine P . Then, we can run A on $D_{\mathbf{y}}$ *without ever evaluating f directly*, and use its answer to determine $f(\mathbf{y})$.

The conditions we impose on \mathcal{A} are quite general, so Theorem 16 has several direct corollaries. For example, recall AC^0 is the class of unbounded fan-in circuits of constant depth over AND, OR, and NOT.

► **Corollary 3.** *The BBH for (polynomial-size) AC^0 analysts is true. Moreover, the BBH for $2^{n^{o(1)}}$ -size AC^0 is true.*

In particular, Theorem 16 implies that for every subexponential-size AC^0 circuit family $\{A_n\}$ that is given the code of an arbitrary (general) circuit C as input, if $\{A_n\}$ computes a semantic property (i.e., its output depends only on the function computed by C , not the code of C) then $\{A_n\}$ must compute a trivial property (all-zeroes or all-ones). Similarly:

Let TC_2^0 be the class of unbounded fan-in circuits of depth-two over MAJORITY, AND, OR, and NOT.

► **Corollary 4.** *The BBH for (polynomial-size) TC_2^0 is true. Moreover, the BBH for $2^{n^{1-\varepsilon}}$ -size TC_2^0 analysts is true for every $\varepsilon > 0$.*

¹ At this level of generality, the idea is similar in spirit to one of the proofs of Rice’s Theorem [19] which shows that any non-trivial semantic property of Turing machines is undecidable, by way of a reduction from the Halting Problem. However, Rice’s proof techniques do not translate to finite circuits, so we prove Theorem 16 differently.

A Generalization

Next, we turn to an even more general setting of Black-Box Hypotheses, where both the class \mathcal{A} of “analysts” and the set \mathcal{C} of “boxes” can vary. Here we find that, roughly speaking, if \mathcal{A} and \mathcal{C} jointly satisfy some natural closure properties, and there are functions computable by boxes in \mathcal{C} but not by analysts in \mathcal{A} , then the \mathcal{C} -Black-Box Hypothesis for \mathcal{A} still holds.

► **Theorem 5** (Informal Statement, cf. Theorem 18 and Theorem 20). *Let \mathcal{A} be a circuit (analyst) class, let \mathcal{C} be a set of circuits, and let $f \notin \mathcal{A}$ be a Boolean function. Suppose there is an analyst in \mathcal{A} which, given input \mathbf{y} , generates a circuit $D_{\mathbf{y}} \in \mathcal{C}$ whose input-output behavior depends on the value of $f(\mathbf{y})$. Then the \mathcal{C} -Black-Box Hypothesis for \mathcal{A} is true.*

We prove two formal versions of this theorem in Section 4.1, as Theorem 18 and Theorem 20. These theorems are general enough that \mathcal{C} does not *have* to be a class of circuits *per se*: other non-uniform computational models, such as branching programs or span programs, would also work. The intuition and proof techniques are similar to those used in Theorem 16, but given the extra conditions on \mathcal{A} , we can tailor the input-switching reduction from Theorem 16 to the set \mathcal{C} in order to produce stronger results. For example:

► **Corollary 6.** *For all primes p , the $\text{AC}^0[p]$ -Black-Box Hypothesis for (poly-size) AC^0 holds.*

Theorem 20 implies that for every AC^0 circuit family $\{A_n\}$ that tries to analyze the code of a given $\text{AC}^0[p]$ circuit C , if $\{A_n\}$ computes a semantic property of C , then that property must be trivial. More generally, we can conclude the following.

► **Theorem 7.** *For all depths $d \geq 2$ and all distinct primes $p \neq q$, the $\text{AC}_d^0[p]$ -Black-Box Hypothesis for $2^{s^{o(1)}}$ -size $\text{AC}^0[q]$ analysts is true.*

That is, even if in the above, $\{A_n\}$ can have subexponential size, use MOD_q gates, and fail on input circuits C with depth greater than a fixed constant d , $\{A_n\}$ must *still* compute a trivial property. Similarly:

► **Theorem 8.** *For all depths $d \geq 2$, the AC_d^0 -Black-Box Hypothesis for $2^{s^{o(1)}}$ -size AC_{d-1}^0 analysts is true.*

Equivalences With Lower Bounds?

So far, our results show how lower bound statements of the form $\mathcal{C} \not\subseteq \mathcal{A}$ can sometimes be applied to prove the corresponding \mathcal{C} -Black-Box Hypothesis for \mathcal{A} analysts. A natural next question is, could Black-Box Hypotheses (for various pairs of boxes and analysts) be *equivalent* to proving lower bounds? As a first step, in Section 5 we prove conditional lower bounds against some analyst classes \mathcal{A} , assuming some \mathcal{C} -Black-Box Hypothesis for \mathcal{A} .

► **Theorem 9** (Informal Statement, cf. Theorem 28). *Suppose every analyst in \mathcal{A} has subexponential-size circuits, and let \mathcal{C} be a “reasonable” set of circuits (left undefined here). If the \mathcal{C} -Black-Box Hypothesis for \mathcal{A} is true, then the circuit satisfiability problem for \mathcal{C} -circuits is not in \mathcal{A} .*

Roughly speaking, we observe that if the \mathcal{C} -circuit Evaluation problem (\mathcal{C} -EVAL) is not in \mathcal{A} , then the \mathcal{C} -Black-Box Hypothesis for \mathcal{A} is **true**, and if the \mathcal{C} -circuit Satisfiability problem (\mathcal{C} -SAT) is in \mathcal{A} , then the \mathcal{C} -Black-Box Hypothesis for \mathcal{A} is **false**. However, \mathcal{C} -SAT is generally harder than \mathcal{C} -EVAL.

To better understand how lower bounds connect to Black-Box Hypotheses, we propose a notion of *BBH-completeness* for computational problems. Very roughly, we want a \mathcal{C} -BBH-complete problem Π to have the property that $\Pi \in \mathcal{C}$, and for a general analyst class \mathcal{A} , if $\Pi \notin \mathcal{A}$ then the \mathcal{C} -BBH for \mathcal{A} is true. We show that for *nondeterministic* circuit classes \mathcal{C} , both \mathcal{C} -SAT and \mathcal{C} -EVAL are \mathcal{C} -BBH-complete.

► **Theorem 10** (Informal Statement, cf. Theorem 31). *Suppose every analyst in \mathcal{A} has subexponential-size circuits, and let \mathcal{C} be a nondeterministic circuit class with “natural” closure properties. Then \mathcal{C} -EVAL and \mathcal{C} -SAT are both \mathcal{C} -BBH-complete for \mathcal{A} .*

Theorem 31 shows that lower bounds for the satisfiability problem are equivalent in some sense to proving that nondeterministic circuits behave like black boxes. Impagliazzo, Kabanets, Kolokolova, McKenzie, and Romani [14] considered the question of whether one can show the Black-Box Hypothesis is equivalent to $\text{NP} \not\subseteq \text{P/poly}$, with some partial results. A consequence of Theorem 31 is that $\text{NP} \not\subseteq \text{P/poly}$ is equivalent to the Black-Box Hypothesis when polynomial-size circuits are the analysts and *nondeterministic circuits* are the boxes. In this light, it would be very interesting if one could show the Black-Box Hypothesis is actually equivalent to $\text{NP} \not\subseteq \text{P/poly}$: it would show that two rather different-looking forms of the Black-Box Hypothesis are in fact equivalent.

Finally, we note that the aforementioned work of Impagliazzo et al. on BBH [14, 20] yields another kind of equivalence between a different variant of black-box hypothesis and a circuit lower bound.

► **Theorem 11** (Follows from [14], informal, cf. Theorem 33). *The following are equivalent:*

1. *The Circuit Satisfiability problem, CKT-SAT, is not in P/poly .*
2. *Any symmetric property P that can be decided in P/poly with white-box access to the input circuit can also be decided in P/poly with black-box access to the input circuit.*

We view this interpretation of their result as further promising evidence towards more general connections between black-box hypotheses and circuit lower bounds.

Organization

Section 2 covers significant prior work related to black-box hypotheses. Section 3 carefully discusses how to generalize the Black-Box Hypothesis for various sets of “analysts” and sets of “boxes”. Section 4 proves our main theorems, showing how circuit lower bounds imply Black-Box Hypotheses in a very generic way. Section 5 considers how we might prove equivalences between Black-Box Hypotheses and lower bounds. Section 6 concludes. The appendices include missing proofs, as well as additional related work.

2 Background

In this paper we assume basic familiarity with computational complexity, especially circuit complexity (knowledge of the first 13 chapters of Arora and Barak [4] would suffice). Throughout the paper, we will recall notation and definitions as needed. Sometimes (as is common in complexity) we will blur the distinction between the analyst class \mathcal{A} as a set of circuit *families* (computing some decision problems) and the actual decision problems computed by analysts in \mathcal{A} .

We will study generic versions of the circuit evaluation and circuit satisfiability problems. In the \mathcal{C} -EVAL problem, we are given a circuit C from a set \mathcal{C} and an input x , and wish to know if $C(x) = 1$. In the \mathcal{C} -SAT problem, we are given a circuit from a set \mathcal{C} and wish to know if there is an x such that $C(x) = 1$. The CKT-SAT problem is \mathcal{C} -SAT where \mathcal{C} is the set of arbitrary Boolean circuits (without loss of generality, each gate has fan-in two).

Historically, researchers interested in so-called “black-box hypotheses” were looking for what they called a “scaled-down” Rice’s Theorem. In the following paragraphs, we provide a brief overview of this research.

Rice’s Theorem

We briefly recall the statement and implications of Rice’s Theorem. Let \mathcal{M} be the set of Turing Machines. We say a property $P : \mathcal{M} \rightarrow \{0, 1\}$ of Turing Machines is *semantic* if $P(M)$ depends only on the (possibly partial) function computed by M . That is, for any TMs M_1 and M_2 with the same input-output behavior, $P(M_1) = P(M_2)$. A property P is *non-trivial* if there are $M_1, M_2 \in \mathcal{M}$ such that $P(M_1) \neq P(M_2)$. In his 1951 doctoral thesis, Henry Rice proved the following sweeping result:

► **Theorem 12** ([19]). *Every non-trivial semantic property of Turing Machines is undecidable.*

Rice’s powerful theorem states that any interesting property that we might want to test of a given program is undecidable, assuming the property being tested depends only on the *function computed by the program*. That is, any property that could in principle be tested using only black-box access to the program, is undecidable given a *description* of the program. Rice’s theorem generalizes (and can be proved from) the undecidability of the TM-SAT problem of determining whether a given TM accepts any string at all.

The Black Box Hypothesis

In their pioneering obfuscation work, Barak et al. [5] consider the question: *can Rice’s Theorem be scaled down in a way that would be useful to complexity theory?* Specifically, let us assume we are not interested in *all* Turing Machines, but rather in the set of efficient algorithms; for example, those represented by Boolean circuits. One can still define properties that are non-trivial and semantic when restricted to the set of Boolean circuits. In this setting, all such properties P are decidable, because the language of a circuit is simply its 2^n -bit truth table, which can be computed in finite time. However, one might want to know something about the *computational complexity* of such properties. In this setting, the circuit satisfiability problem CKT-SAT is an analogue of TM-SAT. Although CKT-SAT is decidable, it is NP-hard, so one might hope to be able to replace undecidability in Rice’s Theorem with NP-hardness.

In earlier work, Borchert and Stephan [8] note that using circuits instead of Turing Machines and NP-hardness instead of undecidability is not enough to prove an analogue of Rice’s Theorem. For every string x , the property $\{M \in \mathcal{M} : M(x) = 1\}$ is undecidable by Rice’s Theorem, but the circuit analogue is decidable in polynomial time: it is simply the circuit evaluation problem! Borchert and Stephan’s response to this issue is to look at function properties depending on more complex measures, such as the *number* of SAT assignments of a given circuit (in other words, the property is a symmetric Boolean function in the truth table of the circuit). They show that any non-trivial “counting” property of circuits is UP-hard; the UP-hardness were improved in [13].

Barak et al. [5] gave a different response to the above issue. They observe the property $\{C : C(x) = 1\}$ for circuits C is still “trivial” in some sense: it can be efficiently determined given only *black-box* oracle access to the input circuit. This observation led Barak et al. to formulate the following conjecture. For two circuits C and C' on n -bit inputs, we write $C \equiv C'$ when C and C' compute the same n -bit function.

► **Conjecture 13** (Black Box Hypothesis [5]). *Suppose $L \subseteq \{0, 1\}^*$ satisfies the property that for all C and C' such that $C \equiv C'$, we have $C \in L \iff C' \in L$. If $L \in \text{BPP}$, then there is a probabilistic polynomial time algorithm S that decides L given only oracle access to C and $0^n 1^{|C|-n}$ as input, i.e.,*

$$C \in L \implies \Pr \left[S^C \left(0^n 1^{|C|-n} \right) = 1 \right] > \frac{2}{3}$$

$$C \notin L \implies \Pr \left[S^C \left(0^n 1^{|C|-n} \right) = 1 \right] < \frac{1}{3}.$$

That is, the BBH claims that every “white box” semantic property of circuits that is decidable in randomized poly-time can also be decided in randomized poly-time with “black box” access to the circuit. If the conjecture were true, then a strong form of $P \neq \text{NP}$ would follow: $P = \text{NP}$ implies that circuit satisfiability is solvable in polynomial-time when we have “white-box” access to the input circuit, but the SAT problem requires $\Omega(2^n)$ time to solve with only black-box oracle access to the input circuit.

Impagliazzo et al. [14] proved interesting results towards understanding BBH. They show a partial converse of the observation from the previous paragraph: if the BBH is false for certain kinds of properties, then the circuit satisfiability problem has sub-exponential size circuits. Since we know that BBH implies $P \neq \text{NP}$, this suggests that it may be difficult to resolve BBH regardless of its truth or falsity. Romani’s master thesis [20] gives an excellent overview of the BBH and this work.

An additional section on “Other Related Work” appears in Appendix A.

3 Generalized Black-Box Hypotheses

We study the Black-Box Hypothesis (Conjecture 13) in a more general setting. Specifically, instead of considering $L \in \text{BPP}$ and a randomized uniform algorithm S from Conjecture 13, we study the family of hypotheses that arise when L and S come from various (possibly non-uniform) circuit classes, which may be weaker or stronger than probabilistic poly-time.

Let us set up some notation. For a circuit C , we let $\langle C \rangle$ denote the *binary description* of C . Note that if C has size s , then $\langle C \rangle$ is a binary string of length $O(s \log s)$, which we call the *description length* of C .

Let \mathcal{C} be a set of circuits. A *property* of circuits in \mathcal{C} is a function $P : \mathcal{C} \rightarrow \{0, 1\}$. A property S is *semantic* iff for any two circuits $C_1, C_2 \in \mathcal{C}$ computing the same function (that is, $\forall \mathbf{x}, C_1(\mathbf{x}) = C_2(\mathbf{x})$), $P(C_1) = P(C_2)$. Recall a circuit family is an infinite sequence of circuits, one for each possible input length; circuit families compute functions of the form $f : \{0, 1\}^* \rightarrow \{0, 1\}$ in the natural way. We say that a circuit family $\{A_s\}$ *computes* P if for every circuit $C \in \mathcal{C}$ with description length s , $A_s(\langle C \rangle) = P(C)$.

We define a **circuit class** \mathcal{A} to be a set of circuit families; our analyst classes \mathcal{A} will have this form. By convention, an *oracle circuit* C may have oracle gates of arbitrary fan-in, but we will think of C as taking an oracle O with a fixed number of inputs. If C contains oracle gates with a different number of inputs than the given oracle O , then we define such oracle gates to output the constant 0 (regardless of O).

We formulate a generalization of the Black Box Hypothesis, which we call the \mathcal{C} -Black Box Hypothesis for \mathcal{A} (\mathcal{C} -BBH for \mathcal{A}), in the following way.

► **Hypothesis 14** (Generalized Black Box Hypothesis: \mathcal{C} -BBH for \mathcal{A}). *Let P be a semantic property of circuits in \mathcal{C} . Let $\{A'_s\} \in \mathcal{A}$ be a circuit family that computes P . Then there exists a circuit family $\{A_s\} \in \mathcal{A}^{\mathcal{C}}$ such that $A_s(1^n 0^{s-n}) = 1$ iff $P(C) = 1$.*

That is, the \mathcal{C} -BBH for \mathcal{A} hypothesizes that every semantic property of \mathcal{C} -circuits that can be decided by \mathcal{A} -analysts with “white box” access to the \mathcal{C} -circuit, can also be decided by \mathcal{A} -analysts with only black-box access to the circuit. When \mathcal{C} is the set of all Boolean circuits, we refer to the \mathcal{C} -BBH for \mathcal{A} simply as the “BBH for \mathcal{A} ”. Note that if we replace \mathcal{A} in the above with BPP, we recover Conjecture 13.

3.1 Encoding Circuits

Unfortunately, if we allow the class of analysts \mathcal{A} to be an arbitrary circuit class, we can encounter some strange (and counterintuitive) consequences. For instance, suppose \mathcal{A} is AC^0 , the circuit families over AND, OR, and NOT with constant-depth, polynomial size, and unbounded fan-in. We can construct an oracle circuit family $\{A_s\}$ such that $A_s^C(1^n 0^{s-n}) = \text{PARITY}(n)$, the parity of the number of inputs of C (A_s^C ignores C , and just computes the parity of strings of the form $1^* 0^*$). Depending on how the description $\langle C \rangle$ is represented, this behavior may not be computable by *any* white-box AC^0 circuit family $\{A'_s\}$, since PARITY is not in AC^0 [1, 9]! We would like to avoid this sort of behavior, because as in Conjecture 13, the oracle circuit family A is supposed to capture some notion of *triviality*. In order for the “BBH for \mathcal{A} ” to be meaningful, it should be that the white-box circuit family A' is at least as powerful as the black-box family A . To this end, we shall require the binary descriptions of circuits to contain all the information given freely to the oracle family. Specifically, we assume that the description of a circuit C with n input wires is prefixed by $1^n 0$, and that the first n wires in $\langle C \rangle$ are the input wires.

4 Circuit Lower Bounds Imply Black-Box Hypotheses

What can we prove about the BBH for general pairs of circuit sets and analysts \mathcal{C} , \mathcal{A} ? First, we can show there are interesting pairs for which the \mathcal{C} -BBH for \mathcal{A} is true in a strong way: every semantic property is in fact trivial. The following theorem shows that, whenever lower bounds hold against a circuit class \mathcal{A} satisfying some simple conditions, the (general) BBH for \mathcal{A} is true. First, we recall a definition.

► **Definition 15.** A projection from n variables onto m variables is a function $\pi : \{0, 1\}^n \rightarrow \{0, 1\}^m$ such that for every j , there exists i such that the j^{th} coordinate of $\pi(\mathbf{x})$ depends only on the i^{th} coordinate of \mathbf{x} .

Observe that a projection is a kind of very weak reduction which can be computed not only very efficiently but also very *locally*. By requiring closure under such a weak class of reductions, we aim to keep \mathcal{A} as general as possible.

► **Theorem 16.** Let \mathcal{A} be a circuit class, $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be a decision problem, and $s : \mathbb{N} \rightarrow \mathbb{N}$ be a monotone function with the properties:

1. f is computable by a size- $s(n)$ circuit family, but f is not computable by any family in \mathcal{A} .
2. Either $\{\text{OR}_n \circ \text{AND}_2\} \subseteq \mathcal{C} \in \mathcal{A}$ for some family \mathcal{C} , or $\{\text{OR}_n \circ \text{AND}_2\} \subseteq \mathcal{C} \in \mathcal{A}$ for some family \mathcal{C} . That is, either \mathcal{A} contains a family that either computes the read-once n -clause 2-DNFs on $2n$ variables, or it contains a family that computes the n -clause 2-CNFs on $2n$ variables.
3. \mathcal{A} is closed under composition with projections from n variables onto $O(s(n) \log s(n))$ variables.

Then for every property P over the set of all circuits, if P is semantic and computable in \mathcal{A} , then for all n , P restricted to circuits on n -bit inputs is also trivial. In particular, the (general) BBH for \mathcal{A} is true.

Proof. Let \mathcal{A} and f satisfy the above properties, and let $\{F_n\}$ be a size- $s(n)$ circuit family computing f . Let P be a semantic property computable in \mathcal{A} .

First, we will prove that P is trivial. The idea is that, if P is not trivial, we can use a circuit family for P to construct a circuit in \mathcal{A} for computing f , a contradiction to the assumed lower bound on f (assumption 1).

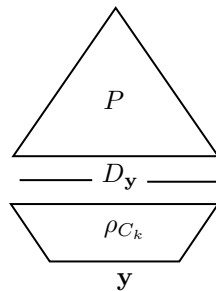
Let $k \in \mathbb{N}$. P is semantic, so assume WLOG that for every k -input circuit K_0 computing the constant 0 function, $P(K_0) = 0$. Assume for sake of contradiction that there is a k -input C_k such that $P(C_k) = 1$. Let $n \in \mathbb{N}$ be our desired input length; we want to build a circuit computing f on n -bit inputs. For an n -bit vector \mathbf{y} , define the following circuit $D_{\mathbf{y}}$ with k input wires \mathbf{x} , with \mathbf{y} hard-coded as n constant wires:

$$D_{\mathbf{y}}(\mathbf{x}) := C_k(\mathbf{x}) \wedge F_n(\mathbf{y}).$$

The circuit $D_{\mathbf{y}}$ computes some Boolean function on k input bits. For a fixed C_k , define the function ρ_{C_k} that maps the n -bit input \mathbf{y} to the description $\langle D_{\mathbf{y}} \rangle$ of $D_{\mathbf{y}}$ as defined above. Observe that for all \mathbf{x} and \mathbf{y} , $D_{\mathbf{y}}(\mathbf{x}) = C_k(\mathbf{x})$ if $f(\mathbf{y}) = 1$, and otherwise $D_{\mathbf{y}}(\mathbf{x}) = 0$. Because P is semantic, $P(D_{\mathbf{y}}) = P(C_k) = 1$ if $f(\mathbf{y}) = 1$, and $P(D_{\mathbf{y}}) = 0$ otherwise. In other words, we have $P(D_{\mathbf{y}}) = f(\mathbf{y})$ for all \mathbf{y} .

Note the size of $D_{\mathbf{y}}$ is $t(k) + \|C_k\| + 1$, where $\|C_k\|$ denotes the size of C_k (which is independent of n), so $D_{\mathbf{y}}$ has description length $O(s(n) \log s(n))$. For a fixed C_k , $\rho_{C_k}(\mathbf{y}) = \langle D_{\mathbf{y}} \rangle$ depends only the n -bit vector \mathbf{y} . In particular, within the description $\langle D_{\mathbf{y}} \rangle$, the descriptions $\langle C_k \rangle$ and $\langle F_n \rangle$ are both independent of \mathbf{y} , so the only bits in $\langle D_{\mathbf{y}} \rangle$ that vary with \mathbf{y} are those describing the hard-coded constant \mathbf{y} itself. Hence each bit in $\langle D_{\mathbf{y}} \rangle$ depends on at most one bit of \mathbf{y} . That is, ρ_{C_k} is a projection from n variables onto $O(s(n) \log s(n))$ variables.

Since \mathcal{A} is closed under such projections (assumption 3), and P is computable in \mathcal{A} by assumption, the circuit



is also computable in \mathcal{A} . However, $P(D_{\mathbf{y}}) = f(\mathbf{y})$, which is not computable in \mathcal{A} , a contradiction. It follows that for all C_k on k inputs, $P(C_k) = 0$, so P (on circuits containing k inputs) is trivial.

We now turn to proving that there exists an oracle circuit family $\{A_s\}$ in \mathcal{A} such that for any circuit C of size s on n inputs, $A_s^C(1^n 0^{s-n}) = P(C)$. In fact we prove the stronger claim that there exists a circuit family $\{A_s\}$ in \mathcal{A} (with no oracle gates) such that for any circuit C of size s on n inputs, $A_s(1^n 0^{s-n}) = P(C)$. To this end, let $X = \{n \in \mathbb{N} : \exists C \text{ on } n \text{ inputs with } P(C) = 1\}$. First, suppose that \mathcal{A} contains a family that can compute $\{\text{OR}_n \circ \text{AND}_2\}$. For $s \in \mathbb{N}$, let A_s be the circuit of the form

$$\bigvee_{i \in X \cap [s]} (x_i \wedge \neg x_{i+1}).$$

29:10 Black-Box Hypotheses and Lower Bounds

By assumptions 2 and 3 (closure under projections from n to $2n$ variables), such circuits are in \mathcal{A} . If instead \mathcal{A} contains $\{\text{AND}_n \circ \text{OR}_2\}$, we let A_s be the circuit of the form

$$\bigwedge_{i \in [s] \setminus X} (\neg x_i \vee x_{i+1}).$$

Now $A_s(1^n 0^{s-n}) = 1$ iff $n \in X$ (using no oracle gates). Since P is trivial, for all circuits C on n inputs, $A_s(1^n 0^{s-n}) = 1$ iff $P(C) = 1$, as desired. ◀

The above proof can be thought of as an “input-switching” trick. We start with the fact that P is non-trivial on some k -bit input circuits. We use the description of a k -input circuit witnessing non-triviality, along with the description of a circuit computing f on n -bit inputs, to construct the description of a larger circuit $D_{\mathbf{y}}$ with n “free variables” \mathbf{y} . By feeding n -bit \mathbf{y} into that description, and feeding that description into P , we obtain the description of an \mathcal{A} -circuit computing f .

Theorem 16 has many immediate corollaries. For example:

► **Reminder of Corollary 3.** *The BBH for (polynomial-size) AC^0 is true. Moreover, the BBH for $2^{n^{o(1)}}$ -size AC^0 is true.*

Proof. Take \mathcal{A} to be AC^0 and f to be the PARITY function in Theorem 16, using the fact that PARITY does not have subexponential-size AC^0 circuits [12]. ◀

► **Reminder of Corollary 4.** *The BBH for (polynomial-size) TC_2^0 is true. Moreover, the BBH for $2^{n^{1-\varepsilon}}$ -size TC_2^0 is true for every $\varepsilon > 0$.*

Proof. Take \mathcal{A} to be TC_2^0 and f to be the INNERPRODUCT function (mod 2) in Theorem 16, using the fact that INNERPRODUCT requires $2^{\Omega(n)}$ -size TC_2^0 circuits [3]. ◀

4.1 Generalization

The proof of Theorem 16 critically relies on the fact that the circuit $D_{\mathbf{y}}$ can be arbitrarily large and complex in comparison to its input. If we restrict \mathcal{C} to contain only “simple” circuits and allow A'_s to behave arbitrarily on circuits not in \mathcal{C} , then we would need to be more careful to ensure that $D_{\mathbf{y}}$ is still in \mathcal{C} . By extending the input-switching trick from Theorem 16, we can restrict the circuit set \mathcal{C} in some interesting ways and still prove the corresponding Black-Box Hypotheses.

► **Definition 17.** *Let \mathcal{C} be a set of circuits, and let f and g be Boolean functions. We say that a function $I : \mathcal{C} \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ is an input-switching function for \mathcal{C} and f iff for some bit b , for every circuit $C \in \mathcal{C}$ and every Boolean string \mathbf{y} , $I(C, \mathbf{y})$ is the description $\langle D_{\mathbf{y}} \rangle$ of a circuit $D_{\mathbf{y}}$ with the same number of inputs as C such that $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ when $f(\mathbf{y}) = b$ and $D_{\mathbf{y}}(\mathbf{x}) = g(\mathbf{x})$ otherwise.*

► **Theorem 18.** *Let \mathcal{A} be a circuit class, $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be a decision problem, and \mathcal{C} be a set of circuits with the properties:*

1. \mathcal{A} computes neither f nor $\neg f$.
2. \mathcal{A} is closed under composition with an input-switching function I for \mathcal{C} and f , in the sense that for every function g computable by a circuit family in \mathcal{A} and for every $C \in \mathcal{C}$, the function $\mathbf{y} \mapsto g(I(C, \mathbf{y}))$ is also computable by a circuit family in \mathcal{A} .

Then for every property P over \mathcal{C} , if P is semantic and computable in \mathcal{A} , then for all input lengths n , P restricted to circuits on n -bit inputs is also trivial. Furthermore, if \mathcal{A} also contains $\{\text{OR}_n \circ \text{AND}_2\}$ (or $\{\text{AND}_n \circ \text{OR}_2\}$), then the \mathcal{C} -BBH for \mathcal{A} is true.

The proof is in Appendix B.

The preconditions for Theorem 18 are somewhat too restrictive to be applied easily in many cases, so we strengthen it further. To this end, we first define a relation \sim_n on sets of circuits.

► **Definition 19.** For sets \mathcal{C}_1 and \mathcal{C}_2 of circuits, say that $\mathcal{C}_1 \sim_n \mathcal{C}_2$ iff there exist n -input circuits $C_1 \in \mathcal{C}_1$ and $C_2 \in \mathcal{C}_2$ such that $C_1 \equiv C_2$ (that is, C_1 and C_2 compute precisely the same Boolean function).

The relation \sim_n enables us to more easily reason about semantic properties across several sets of differently structured circuits.

► **Theorem 20.** Let \mathcal{A} be a circuit class, $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be a decision problem, $\mathcal{C} = \bigcup_{i \in \mathbb{N}} \mathcal{C}_i$ be a set of circuits, and $I : \mathcal{C} \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ a function with the properties:

1. \mathcal{A} computes neither f nor $\neg f$.
2. \mathcal{A} is closed under composition with I .
3. For all i , the restriction of I to $\mathcal{C}_i \times \{0, 1\}^*$ is an input-switching function for \mathcal{C}_i and f .
4. For every input size $n \in \mathbb{N}$, the transitive closure of \sim_n on $\{\mathcal{C}_i\}$ is the universal relation on $\{\mathcal{C}_i\}$.

Then for every property P over \mathcal{C} , if P is semantic and computable in \mathcal{A} , then for all input lengths n , P restricted to circuits on n -bit inputs is also trivial. Furthermore, if \mathcal{A} also contains $\{OR_n \circ AND_2\}$ (or $\{AND_n \circ OR_2\}$), then the \mathcal{C} -BBH for \mathcal{A} is true.

Proof. Let P be a property over \mathcal{C} . Applying Theorem 18 to \mathcal{A} , f , and to each \mathcal{C}_i , for all n and all i , the restrictions of P to circuits in each \mathcal{C}_i with n -bit inputs is trivial. Since P is semantic, if $i \sim_n j$, then the restriction of P to circuits in $\mathcal{C}_i \cup \mathcal{C}_j$ with n -bit inputs is also trivial. Finally since the transitive closure of \sim_n is universal, by induction we have that for every n , the restriction of P to circuits in \mathcal{C} with n -bit inputs is trivial. ◀

4.2 Examples

We now define some input-switching functions. First, let f be any function computable by a circuit family $\{F_n\}$, and let $D_{\mathcal{C}, \mathbf{y}}$ be the circuit defined as follows, where \mathbf{x} are the input wires and \mathbf{y} are hard-coded as n constant wires:

$$D_{\mathcal{C}, \mathbf{y}}(\mathbf{x}) := C(\mathbf{x}) \wedge F_n(\mathbf{y}).$$

If F_n has size $s(n)$, then the map $\mathbf{y} \mapsto \langle D_{\mathcal{C}, \mathbf{y}} \rangle$ (where $\langle D_{\mathcal{C}, \mathbf{y}} \rangle$ is the description of $D_{\mathcal{C}, \mathbf{y}}$) is both an input-switching function and a projection from n variables onto the $O(s(n) \log s(n))$ variables describing $D_{\mathcal{C}, \mathbf{y}}$, so we recover Theorem 16.

Recall that $AC_d^0[p]$ denotes circuit families of depth d with unbounded fan-in AND, OR, and MOD_p gates.

► **Reminder of Corollary 6.** For all primes p , the $AC^0[p]$ -Black-Box Hypothesis for (polynomial-size) AC^0 is true. Moreover, the $AC^0[p]$ -Black-Box Hypothesis for $2^{s^{o(1)}}$ -size AC^0 is true.

Proof. Follows from Theorem 18. We make use of the fact that the MOD_p function is computable in linear size $AC^0[p]$ but requires exponential size in AC^0 , and that in AC^0 we can mask a given $AC^0[p]$ circuit with a given MOD_p function.

29:12 Black-Box Hypotheses and Lower Bounds

Let \mathcal{A} be $2^{s^{o(1)}}$ -size AC^0 , $f = \text{MOD}_p$, and $\mathcal{C} = \text{AC}^0[p]$. We now define a circuit $D_{C,\mathbf{y}}$ with the same number of inputs as C as

$$D_{C,\mathbf{y}}(\mathbf{x}) := C(\mathbf{x}) \wedge \text{MOD}_p(\mathbf{y}).$$

Then for all \mathbf{x} and \mathbf{y} , $D_{C,\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $\text{MOD}_p(\mathbf{y}) = 1$, and $D_{C,\mathbf{y}}(\mathbf{x}) = 0$ otherwise. Now the map $(C, \mathbf{y}) \mapsto \langle D_{C,\mathbf{y}} \rangle$ is an input-switching function for \mathcal{C} and MOD_p . Furthermore, we can think of the map $\mathbf{y} \mapsto \langle D_{C,\mathbf{y}} \rangle$ as a projection from n variables \mathbf{y} onto $\Theta(n \log n)$ variables describing $D_{C,\mathbf{y}}$, so \mathcal{A} is closed under composition with I . Now from Theorem 18, every semantic property P over \mathcal{C} computable in \mathcal{A} is trivial, so the \mathcal{C} -BBH for \mathcal{A} is true. \blacktriangleleft

If we invoke Theorem 20 instead of Theorem 18, we can get an even stronger result.

► **Theorem 21.** *For all depths $d \geq 2$ and distinct primes $p \neq q$, the $\text{AC}_d^0[p]$ -BBH for $2^{s^{o(1)}}$ -size $\text{AC}^0[q]$ is true.*

The proof is in Appendix C. The proof of Theorem 21 relies on the fact that small $\text{AC}^0[q]$ circuits cannot evaluate some function that can be evaluated with small $\text{AC}^0[p]$ circuits (namely a single MOD_p gate). We can prove a similar result using the depth- d Sipser function, which is easy for AC^0 circuits of depth d but hard for depth $d - 1$ [22, 12].

► **Definition 22.** *The Sipser function $f^{d,n} : \{0, 1\}^{\sqrt{\frac{n}{\log n}}} \times \{0, 1\}^{n^{d-2}} \times \{0, 1\}^{\sqrt{\frac{1}{2}dn \log n}} \rightarrow \{0, 1\}$ is defined as follows:*

$$\begin{aligned} \text{If } d \text{ is odd, then } f^{d,n}(\mathbf{x}) &= \bigwedge_{i_1=1}^{\sqrt{\frac{n}{\log n}}} \bigvee_{i_2=1}^n \bigwedge_{i_3=1}^n \cdots \bigwedge_{i_d=1}^{\sqrt{\frac{1}{2}dn \log n}} x_{i_1, \dots, i_d}. \\ \text{If } d \text{ is even, then } f^{d,n}(\mathbf{x}) &= \bigwedge_{i_1=1}^{\sqrt{\frac{n}{\log n}}} \bigvee_{i_2=1}^n \bigwedge_{i_3=1}^n \cdots \bigvee_{i_d=1}^{\sqrt{\frac{1}{2}dn \log n}} x_{i_1, \dots, i_d}. \end{aligned}$$

► **Theorem 23.** *For all depths $d \geq 2$, the AC_d^0 -BBH for $2^{s^{o(1)}}$ -size AC_{d-1}^0 is true.*

The proof is in Appendix D.

5 Some Black-Box Hypotheses Imply Lower Bounds

In Section 4, we showed that many circuit lower bounds of the form $\mathcal{C}' \not\subseteq \mathcal{A}$ can be used to prove a corresponding \mathcal{C} -Black-Box Hypothesis for \mathcal{A} (for a set of boxes \mathcal{C} that suitably captures the complexity class \mathcal{C}'). Now we consider the converse question: can Black-Box Hypotheses also be used to prove circuit lower bounds? For certain sets \mathcal{C} of boxes and classes \mathcal{A} of analysts, it turns out that the \mathcal{C} -Black-Box Hypothesis for \mathcal{A} does in fact imply lower bounds against \mathcal{A} .

For a function $s : \mathbb{N} \rightarrow \mathbb{N}$, let $\text{CIRCUIT}(s(n))$ denote the set of (general) Boolean circuits on n inputs of size at most $s(n)$, for every n . (Note this is different from $\text{SIZE}(s(n))$, which is the class of *languages* computed by circuit families of size at most $s(n)$.) As a starting point, the following simple proposition was essentially noted by Barak et al. [5].

► **Proposition 24.** *If $\text{NP} \subset \text{P}/\text{poly}$, then for every polynomial p , the $\text{CIRCUIT}(p(n))$ -BBH for P/poly is false.*

Proof. Take P to be the CKT-SAT property (that is, $P(C) = 0$ iff the circuit C encodes the all-zeroes function). By assumption, $P \in \text{P/poly}$, but even with randomness, $\Omega(2^n)$ oracle queries are needed to determine whether a size- $p(n)$ circuit on n inputs is the all-zeroes function. For every polynomial q , the polynomial $q \circ p$ is $o(2^n)$, so there is no size- $q(s)$ circuit family making $\Omega(2^n)$ oracle queries on size- $p(n)$ circuits. ◀

In fact, Proposition 24 can be strengthened by replacing CKT-SAT with the property $P(C) = 1$ iff C has a satisfying assignment that sets the first k inputs to 0 (for some appropriately large k).

► **Proposition 25.** *If $\text{NP} \subset \text{SIZE}(2^{n^{o(1)}})$, then for every polynomial p , the $\text{CIRCUIT}(p(n))$ -BBH for P/poly is false.*

Propositions 24 and 25 are arguably not particularly useful, since very few researchers believe the hypotheses of these propositions. However, they still do illustrate an interesting observation, and we may be able to generalize them in a useful manner. Let \mathcal{C} -SAT be the satisfiability problem for circuits from the set \mathcal{C} . One might hope to prove the following generalization of Proposition 24, for every circuit set \mathcal{C} and every analyst class \mathcal{A} :

► **Hypothesis 26** (The Satisfiability Black-Box Hypothesis). *If $\mathcal{C}\text{-SAT} \in \mathcal{A}$, then the \mathcal{C} -BBH for \mathcal{A} is false.*

In this fully generic form, there are some simple counterexamples to Hypothesis 26. For instance, if \mathcal{A} contains *all* Boolean functions, then (for every set \mathcal{C}) $\mathcal{C}\text{-SAT} \in \mathcal{A}$. However, the \mathcal{C} -BBH for \mathcal{A} is *true*, because \mathcal{A} can decide *any* semantic property with only black-box access to the circuit being analyzed. Hence we require additional restrictions on \mathcal{C} and \mathcal{A} to make the hypothesis interesting. In particular, we would like \mathcal{A} to contain only functions of subexponential circuit complexity, and for a sufficiently simple function f , we would like \mathcal{C} circuits to be able to compute f efficiently.

Recall that a Boolean function $f : \{0,1\}^* \rightarrow \{0,1\}$ is a *point function* if there is an $\mathbf{a} \in \{0,1\}^*$ such that for all \mathbf{x} , $f(\mathbf{x}) = 1 \iff \mathbf{x} = \mathbf{a}$. The following notion of “reasonability” for circuit sets will be useful in multiple contexts.

► **Definition 27** (Reasonability). *A set \mathcal{C} of circuits is reasonable if there is a polynomial p such that for all point functions f , there is a circuit family $\{C_n\} \subset \mathcal{C}$ of size at most $p(n)$ computing f .*

We can show that if \mathcal{C} is reasonable and \mathcal{A} has subexponential-size circuits, then Hypothesis 26 is true. The following can be viewed as a kind of converse of Theorem 16.

► **Theorem 28.** *If \mathcal{C} is reasonable, $\mathcal{A} \subseteq \text{SIZE}(2^{n^{o(1)}})$, and $\mathcal{C}\text{-SAT} \in \mathcal{A}$, then the \mathcal{C} -BBH for \mathcal{A} is false.*

Proof. Assume \mathcal{C} is reasonable, \mathcal{A} has subexponential-size circuits, and $\mathcal{C}\text{-SAT} \in \mathcal{A}$. As in Proposition 24, we take P to be the satisfiability property. By assumption, $P \in \mathcal{A}$. Even with randomness, $\Omega(2^n)$ oracle queries are required to determine whether a circuit of size $p(n)$ on n inputs computes the constant 0 function. However, an \mathcal{A} circuit can make at most $2^{n^{o(1)}}$ queries to its oracle when given an input of size $p(n)$. Per the reasonableness of \mathcal{C} , there are both satisfiable and unsatisfiable \mathcal{C} -circuits of size $p(n)$, so \mathcal{A} , with only black-box access to a \mathcal{C} -circuit, cannot compute P . ◀

The preconditions for Theorem 28 are very general; most complexity classes of interest only deal with functions of subexponential complexity and can compute point functions efficiently. However, this weak condition is sufficient to remove the simple counterexamples.

5.1 A Notion of BBH-Completeness

For very general circuit sets \mathcal{C} and classes \mathcal{A} of analysts, we have shown (roughly) in Section 4 that

$$\mathcal{C} \not\subseteq \mathcal{A} \implies \mathcal{C}\text{-BBH for } \mathcal{A},$$

and in the previous paragraphs that for “reasonable” \mathcal{A} and \mathcal{C} ,

$$\mathcal{C}\text{-BBH for } \mathcal{A} \implies \mathcal{C}\text{-SAT} \not\subseteq \mathcal{A}.$$

For many pairs of classes \mathcal{C} and \mathcal{A} , we have

$$\mathcal{C}\text{-EVAL} \not\subseteq \mathcal{A} \iff \mathcal{C} \not\subseteq \mathcal{A}.$$

So the results of Section 4 imply, at least for many natural pairs \mathcal{C}, \mathcal{A} , that \mathcal{C} -EVAL lower bounds imply BBHs. However, \mathcal{C} -SAT is generally a harder problem than \mathcal{C} -EVAL, so there remains a gap between the lower bounds that provably imply a Black-Box Hypothesis, and those lower bounds provably implied by a Black-Box Hypothesis.

A natural question is then, which of these implications can be strengthened? **Is there a single problem on \mathcal{C} circuits, such that proving a lower bound for it is equivalent to proving a \mathcal{C} -Black-Box Hypothesis?** In particular, is proving either $\mathcal{C}\text{-EVAL} \not\subseteq \mathcal{A}$ or $\mathcal{C}\text{-SAT} \not\subseteq \mathcal{A}$ *equivalent* to proving the \mathcal{C} -BBH for \mathcal{A} ? Similar to other completeness notions in complexity theory, we propose a concept of BBH-completeness to study equivalences between circuit lower bounds and Black-Box Hypotheses.²

► **Definition 29** (BBH-completeness). *Let \mathcal{C} be a set of circuits and \mathcal{A} a complexity class. A Boolean function f is complete for the \mathcal{C} -BBH for \mathcal{A} (or \mathcal{C} -BBH-complete for \mathcal{A}) iff*

$$\mathcal{C}\text{-BBH for } \mathcal{A} \iff f \notin \mathcal{A}.$$

When \mathcal{A} is either implicitly understood or general, we say that f is \mathcal{C} -BBH-complete.

Are there natural pairs $(\mathcal{C}, \mathcal{A})$ for which either \mathcal{C} -EVAL or \mathcal{C} -SAT is \mathcal{C} -BBH-complete for \mathcal{A} ?

The Case of Nondeterministic Boxes. For the case of sets \mathcal{C} of nondeterministic circuits, the answer is **yes**. To state our theorem, we require one new concept. Recall that a nondeterministic circuit C has a sequence of “normal” inputs \mathbf{x} as well as a sequence of “auxiliary” nondeterministic inputs \mathbf{y} , and we say that C accepts \mathbf{x} if there is a setting of \mathbf{y} such that $C(\mathbf{x}, \mathbf{y}) = 1$.

► **Definition 30.** *For a given circuit C , a nondeterminization of C is a circuit C' in which normal inputs to C have been converted into auxiliary nondeterministic inputs. A set \mathcal{C} of circuits is closed under nondeterminization if $C \in \mathcal{C}$ implies that every nondeterminization of C is also in \mathcal{C} .*

² It must be said that both authors are not entirely comfortable with the following definition of BBH-completeness. Ideally, the following would be a consequence of f being BBH-complete, and the actual definition would involve a notion of reducibility. However, in order to give a completeness concept that fits all possible classes \mathcal{A} and \mathcal{C} at a high level of generality, it does not seem possible to use reductions: a sound reducibility notion would inevitably have to depend on \mathcal{A} (in particular, its allowed “sizes” and its closure properties) directly.

► **Theorem 31.** *Let \mathcal{C} be a reasonable set of circuits closed under nondeterminization. Assume \mathcal{A} has circuits of size $2^{n^{o(1)}}$ and that \mathcal{A} is closed under composition with an input-switching function for \mathcal{C} and \mathcal{C} -EVAL. Then \mathcal{C} -EVAL and \mathcal{C} -SAT are \mathcal{C} -BBH-complete for \mathcal{A} .*

Proof. We wish to prove that the following are equivalent:

1. \mathcal{C} -BBH for \mathcal{A}
2. \mathcal{C} -SAT $\notin \mathcal{A}$
3. \mathcal{C} -EVAL $\notin \mathcal{A}$

(1) \implies (2) Follows from Theorem 28.

(2) \implies (3) We reduce \mathcal{C} -SAT to \mathcal{C} -EVAL by observing that changing the inputs of a nondeterministic circuit into auxiliary nondeterministic inputs preserves satisfiability. Hence, given a nondeterministic circuit C , we can convert *all* of its input bits into additional nondeterministic auxiliary inputs to obtain a circuit C' , and then determine whether C' is still satisfiable. However, C' has no remaining free inputs, so determining satisfiability of C' is simply the problem of *evaluating* C' (with no inputs).

(3) \implies (1) Follows from Theorem 18. ◀

Interpreting Impagliazzo et al. as an Equivalence. Recently, Impagliazzo et al. [14] proved that if the BBH is false for certain kinds of function properties, then the circuit satisfiability problem has sub-exponential size circuits. In particular, they show that CKT-SAT has $2^{n^{o(1)}}$ -size circuits if a property P is highly sensitive on a function f that has sub-exponential size circuits.

Impagliazzo et al. indicate that in some sense CKT-SAT is BBH-complete, at least for large analyst classes \mathcal{A} . Specifically, if we consider only *symmetric* semantic properties, i.e., properties that depend only on the number of ones in the truth table of the input circuit, we can define the following conjecture:

► **Hypothesis 32 (Symmetric-BBH).** *Let P be a semantic and symmetric property of circuits. Let $\{A'_s\}$ be a polynomial size circuit family. Assume that for every circuit C of size s on n inputs, $A'_s(C) = 1$ iff $P(C) = 1$. Then there exists a polynomial size oracle circuit family $\{A_s\}$ such that $A_s^C(1^n 0^{s-n}) = 1$ iff $P(C) = 1$.*

Now [14] implies:

► **Theorem 33 (Follows from [14]).** *The following are equivalent:*

1. CKT-SAT is not in P/poly.
2. The Symmetric-BBH is true.

Proof. The forward direction is Corollary 4.3 in [14]. For the converse direction, observe that CKT-SAT is a symmetric property that requires exponentially many black-box oracle queries (and in particular, cannot be solved in P/poly with only black-box access to the input circuit). Hence if the Symmetric-BBH is true, then CKT-SAT also cannot be solved in P/poly with white-box access to the input circuit, i.e., CKT-SAT \notin P/poly. ◀

6 Conclusion

In this paper, we introduced *generalized* Black-Box Hypotheses, which parameterize both the type of “box” being analyzed, and the type of “analyst” examining such boxes. We showed that generalized Black-Box Hypotheses can follow generically from circuit lower bounds, and we showed how lower bounds for the circuit satisfiability problem are essentially equivalent to Black-Box Hypotheses where the “boxes” correspond to nondeterministic circuits. We conclude with some additional interesting directions to consider.

What Other Lower Bounds Are Implied by Black-Box Hypotheses? In Section 5 we noted a simple example of a lower bound implying a BBH: the \mathcal{C} -BBH for \mathcal{A} implies \mathcal{C} -SAT $\notin \mathcal{A}$. However, this lower bound is rather weak-looking: \mathcal{C} -SAT is NP-complete for many very simple \mathcal{C} . Are there circuit-analysis problems which are likely *not* to be NP-complete, which would still be implied by a Black-Box Hypothesis? We find this to be a very interesting question, and we currently do not have good candidates for such a problem.

Randomized Lower Bounds and Their Black-Box Hypotheses. We have shown that (deterministic) worst-case lower bounds can lead to results about analyzing circuits as boxes. What results can be derived from *average-case* or *randomized* lower bounds? We have obtained some preliminary results in this direction. For instance, if our analyst class \mathcal{A} consists of *randomized* algorithms rather than deterministic ones, we can still prove connections between lower bounds against \mathcal{A} and BBHs for \mathcal{A} , along the lines of Section 4. There are likely other connections like this to be found within the vast landscape of complexity theory.

Black-Box Hypotheses From More Lower Bounds? While we have shown that various Black-Box Hypotheses do follow from certain lower bounds in a generic way, some lower bounds don't seem to imply a Black-Box Hypothesis. For example, a circuit-size hierarchy is well-known: for nice functions s , there are functions computable with size- $s(n)$ circuits that do not have circuits of size less than $s(n) - 5n$ (cf. [15]). This suggests the possibility that, for analysts \mathcal{A} implemented by circuits of size less than $s(n) - 5n$, and boxes \mathcal{C} which are circuits of size at least $s(n)$, the \mathcal{C} -Black-Box Hypothesis for \mathcal{A} is true. However, our current methods are unable to prove such a sharp result. Are there other intermediate lower bounds (weaker than against e.g. \mathcal{C} -EVAL) that would still imply Black-Box Hypotheses?

References

- 1 Miklos Ajtai. Σ_1^1 -formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.
- 2 Shaul Almagor, Brynmor Chapman, Mehran Hosseini, Joël Ouaknine, and James Worrell. Effective divergence analysis for linear recurrence sequences. In *Proceedings of the 29th International Conference on Concurrency Theory (CONCUR 2018), LIPIcs 118*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- 3 Kazuyuki Amano. On the size of depth-two threshold circuits for the inner product mod 2 function. In Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, editors, *Language and Automata Theory and Applications*, pages 235–247, Cham, 2020. Springer International Publishing.
- 4 Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009. doi:10.1017/CB09780511804090.
- 5 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual international cryptography conference*, pages 1–18. Springer, 2001.
- 6 Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM (JACM)*, 65(6):39, 2018.
- 7 Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *Algorithmica*, 79(4):1233–1285, 2017.
- 8 Bernd Borchert and Frank Stephan. Looking for an analogue of Rice's theorem in circuit complexity theory. In *Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pages 114–127. Springer, 1997.

- 9 Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. See also FOCS’81.
- 10 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.
- 11 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Hiding secrets in software: A cryptographic approach to program obfuscation. *Communications of the ACM*, 59(5):113–120, 2016.
- 12 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM symposium on Theory of computing*, pages 6–20, 1986.
- 13 Lane A Hemaspaandra and Mayur Thakur. Lower bounds and the hardness of counting properties. *Theoretical computer science*, 326(1-3):1–28, 2004.
- 14 Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, Pierre McKenzie, and Shadab Romani. Does looking inside a circuit help? In *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 15 Stasys Jukna. *Boolean Function Complexity – Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.
- 16 Joël Ouaknine and James Worrell. Decision problems for linear recurrence sequences. In *Proceedings of the 6th International Workshop on Reachability Problems (RP 2012)*, LNCS 7550. Springer, 2012.
- 17 Joël Ouaknine and James Worrell. Ultimate positivity is decidable for simple linear recurrence sequences. In *Proceedings of 41st International Colloquium on Automata, Languages, and Programming (ICALP 2014)*, LNCS 8573. Springer, 2014.
- 18 Alexander Razborov. Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.
- 19 Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953.
- 20 Shadab Romani. *Succinct representations of Boolean functions and the Circuit-SAT problem*. PhD thesis, Memorial University of Newfoundland, 2016.
- 21 Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 2014.
- 22 Michael Sipser. Borel sets and circuit complexity. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 61–69, 1983. doi:10.1145/800061.808733.
- 23 Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *STOC*, pages 77–82, 1987.

A Other Related Work

Obfuscation in Cryptography

In recent years, the theory of program obfuscation has exploded into a huge subject area within cryptography, starting with an influential paper of Barak et al. [5] which crystallized several key definitions and proved key impossibility results for obfuscation. Two major concepts they proposed are *virtual black-box obfuscation* (VBB for short) and *indistinguishability obfuscation* (iO for short), which we now describe briefly.

A VBB obfuscator \mathcal{O} would take any efficient program/circuit C of size s , and output the code of an “obfuscated” $\mathcal{O}(C)$ such that, for every probabilistic polynomial time (PPT) adversary A , there is another PPT adversary A' , such that the probability A outputs 1 on

the input $\mathcal{O}(C)$ is very close to the probability that A' outputs 1 on $(1^n, 1^s)$ when *given* C as an oracle. That is, whatever computation A is doing on the code of $\mathcal{O}(C)$, A' can simulate that knowing only the size of C , its number of inputs, and with input-output access to C . Barak et al. showed that there are tasks for which VBB obfuscation is *impossible* assuming one-way functions exist. The notion of iO asks for a weaker guarantee: for all PPT A , and all pairs of size- s circuits C_1, C_2 such that $C_1 \equiv C_2$, the probability A outputs 1 on C_1 is very close to the probability A outputs 1 on C_2 . In contrast to VBB, iO is possible under plausible hardness conjectures (e.g., [11, 10, 6]), and it turns out to be very powerful, capable of implementing deniable encryption, public-key encryption from one-way functions, multiparty key exchange, and more (e.g., [21, 7]).

All of the above work on building obfuscation requires hardness assumptions that are unproved (and are typically much stronger than $P \neq NP$), and study how we might efficiently transform arbitrary code into obfuscated code, relative to some class of adversarial analysts.

We briefly note the connection between VBB and the BBH. One can think of a VBB obfuscator as an efficient mapping from general circuits to “obfuscated class of circuits”, a *restricted subclass* of circuits, such that the BBH holds when the analyzable code \mathcal{C} must come from this restricted subclass. Namely, the VBB property says that, for any efficient analyst that takes circuits from this class as input, there is an efficient black-box analyst that can carry out essentially the same analyses. That is, when VBB is possible, there is a “promise” class of circuits (the image of the obfuscator) for which a black-box hypothesis is true. Accordingly, Barak et al. [5] showed that a “promise” version of the BBH is false, assuming one-way functions exist.

Automated Formal Verification

Additionally, settings in which the Black-Box Hypothesis is false are of great interest in automated formal verification. One central question is the following: what properties of a program’s input-output behavior can be more efficiently tested by analyzing the program’s code, than by treating it as a black box and simply running it on selected inputs? Many properties of interest depend on the program’s behavior on all possible inputs, which may be infeasible (or even impossible) to determine exhaustively. One may instead want to analyze the code of the program in order to determine whether or not it satisfies the given property. This may still be impossible, as many properties of interest are Turing-complete when considered over the space of all possible programs. However, by restricting the class of programs being tested, some such verification problems can become feasible, cf. [16, 17, 2]. In fact, in any setting where the class of programs being analyzed is restricted such that the black box hypothesis is *false*, there *must* exist properties that can be tested by analyzing the program but not by treating it as a black box.

B Proof of Theorem 18

► **Reminder of Theorem 18.** Let \mathcal{A} be a circuit class, $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be a decision problem, and \mathcal{C} be a set of circuits with the properties:

1. \mathcal{A} contains neither f nor $\neg f$.
2. \mathcal{A} is closed under composition with an input-switching function I for \mathcal{C} and f , in the sense that for every function g computable by a circuit family in \mathcal{A} and for every $C \in \mathcal{C}$, the function $\mathbf{y} \mapsto g(I(C, \mathbf{y}))$ is also computable by a circuit family in \mathcal{A} .

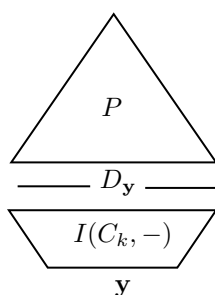
Then for every property P over \mathcal{C} , if P is semantic and computable in \mathcal{A} , then for all input lengths n , P restricted to circuits on n -bit inputs is also trivial. Furthermore, if \mathcal{A} also contains $\{\text{OR}_n \circ \text{AND}_2\}$ (or $\{\text{AND}_n \circ \text{OR}_2\}$), then the \mathcal{C} -BBH for \mathcal{A} is true.

Proof. Let \mathcal{A} and f satisfy the above properties, and let I be the input-switching function for \mathcal{C} and f . Let P be a semantic property computable in \mathcal{A} .

First, we will prove that P is trivial. The idea is that, if P is not trivial, we can use a circuit family for P to construct a circuit in \mathcal{A} for computing f , a contradiction to the assumed lower bound on f .

Let $k \in \mathbb{N}$. Assume WLOG that for every circuit G computing g on k inputs, $P(G) = 0$. Assume for sake of contradiction that there is a k -input C_k such that $P(C_k) = 1$. For an n -bit input \mathbf{y} , consider the circuit $D_{\mathbf{y}} = I(C_k, y)$. Note that $D_{\mathbf{y}}$ computes some Boolean function on k input bits. From the definition of I , $D_{\mathbf{y}}(\mathbf{x}) = C_k(\mathbf{x})$ if $f(\mathbf{y}) = b$, and otherwise $D_{\mathbf{y}}(\mathbf{x}) = G(\mathbf{x})$. Because P is semantic, $P(D_{\mathbf{y}}) = P(C_k) = 1$ if $f(\mathbf{y}) = b$, and $P(D_{\mathbf{y}}) = P(G) = 0$ otherwise. In other words, we have $P(D_{\mathbf{y}}) = b \otimes f(\mathbf{y})$ for all \mathbf{y} .

Since \mathcal{A} is closed under composition with $I(C_k, -)$, and P is computable in \mathcal{A} by assumption, the circuit



is also computable in \mathcal{A} . However, $P(D_{\mathbf{y}}) = f(\mathbf{y})$ or $\neg f(\mathbf{y})$, which are not computable in \mathcal{A} , a contradiction. It follows that for all C_k on k inputs, $P(C_k) = 0$, so P (on circuits containing k inputs) is trivial.

As in Theorem 16, there exists a circuit family $\{A_s\}$ in \mathcal{A} (with no oracle gates) such that for any circuit C of size s on n inputs, $A_s(1^n 0^{s-n}) = P(C)$. ◀

C Proof of Theorem 21

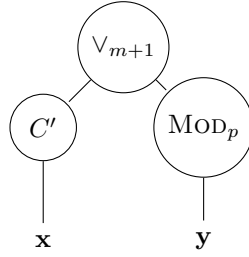
▶ **Reminder of Theorem 21.** For all depths $d \geq 2$ and distinct primes $p \neq q$, the $\text{AC}_d^0[p]$ -BBH for $2^{s^{o(1)}}$ -size $\text{AC}^0[q]$ is true.

Proof. Follows from Theorem 20. We make use of the fact that the MOD_p function is computable in linear size $\text{AC}_d^0[p]$ but requires exponential size in $\text{AC}^0[q]$ [18, 23], and that in AC^0 we can mask a given $\text{AC}_d^0[p]$ circuit with a given MOD_p function, without increasing its depth.

Let $d \geq 2$, \mathcal{A} be $2^{s^{o(1)}}$ -size $\text{AC}^0[q]$, $f = \text{MOD}_p$, $\mathcal{C} = \text{AC}_d^0[p]$, $\mathcal{C}_1 = \text{OR} \circ \text{AC}_{d-1}^0[p]$, $\mathcal{C}_2 = \text{AND} \circ \text{AC}_{d-1}^0[p]$, and $\mathcal{C}_3 = \text{MOD}_p \circ \text{AC}_{d-1}^0[p]$. (Note that $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3$.) We now define a function $I : \mathcal{C} \times \{0, 1\}^* \rightarrow \{0, 1\}^*$, so that $I(C, \mathbf{y}) = \langle D_{\mathbf{y}} \rangle$, where $D_{\mathbf{y}}$ has the same number of inputs as C . We condition on whether the input circuit C comes from \mathcal{C}_1 , \mathcal{C}_2 , or \mathcal{C}_3 .

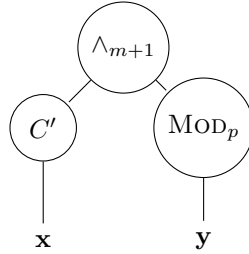
Case 1. If $C \in \mathcal{C}_1$, then it has the form $\vee_m \circ C'$, where C' is a depth- $(d-1)$ circuit with n inputs and m outputs, and \vee_m is an OR of fan-in m . For a k -bit vector \mathbf{y} , we construct $D_{\mathbf{y}}$ as follows:

29:20 Black-Box Hypotheses and Lower Bounds



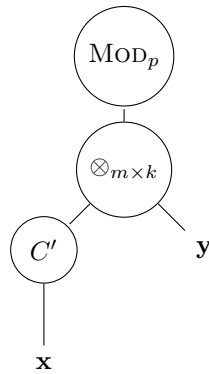
Then for all \mathbf{x} and \mathbf{y} , $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $\text{MOD}_p(\mathbf{y}) = 0$, and $D_{\mathbf{y}}(\mathbf{x}) = 1$ otherwise. Hence the restriction of I to $\mathcal{C}_1 \times \{0, 1\}^*$ is an input-switching function for \mathcal{C}_1 and f .

Case 2. If $C \in \mathcal{C}_2$, we construct $D_{\mathbf{y}}$ similarly to case (1). Assuming $C = \wedge_m \circ C'$ for the same sort of C' , we can construct $D_{\mathbf{y}}$ as follows:



Then for all \mathbf{x} and \mathbf{y} , $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $\text{MOD}_p(\mathbf{y}) = 1$, and $D_{\mathbf{y}}(\mathbf{x}) = 0$ otherwise. Hence the restriction of I to $\mathcal{C}_2 \times \{0, 1\}^*$ is an input-switching function for \mathcal{C}_2 and f .

Case 3. If $C \in \mathcal{C}_3$, then C is a MOD_p gate of fan-in m , composed with some depth- $(d - 1)$ circuit C' having n inputs and m outputs. We define a $\otimes_{m \times k}$ gate to take $m + k$ inputs $x_1, \dots, x_m, y_1, \dots, y_k$, and output $x_i \cdot y_j$ for all i, j , and define a circuit $D'_{\mathbf{y}}(\mathbf{x})$ as follows:



Note that for C of depth d , $D'_{\mathbf{y}}$ has depth $d + 1$. However, when treating \mathbf{y} as a constant, each $C'(\mathbf{x})_i \wedge y_j$ simplifies to a single wire (either $C'(\mathbf{x})_i$ if $y_j = 1$, or the constant 0 if $y_j = 0$). Performing these simplifications and removing the layer of AND gates, we get a circuit $D_{\mathbf{y}}$ of depth d . (Note that each bit in $\langle D_{\mathbf{y}} \rangle$ still only depends on at most one bit of \mathbf{y} .) Now for all \mathbf{x} and \mathbf{y} , $D_{\mathbf{y}}(\mathbf{x}) = \text{MOD}_p(C'(\mathbf{x}) \otimes \mathbf{y}) = \text{MOD}_p(C'(\mathbf{x})) \times \text{MOD}_p(\mathbf{y}) = C(\mathbf{x}) \times \text{MOD}_p(\mathbf{y})$. That is, $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $\text{MOD}_p(\mathbf{y}) = 1$, and $D_{\mathbf{y}}(\mathbf{x}) = 0$ otherwise. Hence the restriction of I to $\mathcal{C}_3 \times \{0, 1\}^*$ is an input-switching function for \mathcal{C}_3 and f .

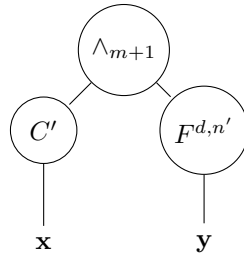
Next, we observe that in every case, each bit in $\langle D_{\mathbf{y}} \rangle$ depends on only one bit of \mathbf{y} , so \mathcal{A} is closed under composition with I (a projection). Finally, there are circuits C_1, C_2, C_3 , which have an OR, AND, and MOD_p output gate (respectively), yet $C_1 \equiv C_2 \equiv C_3$ (e.g. they can ignore their input and output the constant 0). Hence \sim_k as defined in Theorem 20 is the universal relation. Now from Theorem 20, every semantic property P over \mathcal{C} computable in \mathcal{A} is trivial, so the \mathcal{C} -BBH for \mathcal{A} is true. ◀

D Proof of Theorem 23

► **Reminder of Theorem 23.** For all depths $d \geq 2$, the AC_d^0 -BBH for $2^{s^{o(1)}}$ -size AC_{d-1}^0 is true.

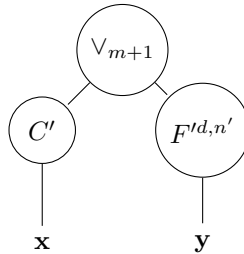
Proof. We proceed as in Theorem 21. Let $d \geq 2$, f be the depth- d Sipser function, \mathcal{A} be $2^{s^{o(1)}}$ -size AC_{d-1}^0 , $\mathcal{C} = \text{AC}_d^0$, $\mathcal{C}_1 = \text{AND} \circ \text{AC}_{d-1}^0$, and $\mathcal{C}_2 = \text{OR} \circ \text{AC}_{d-1}^0$. (Note that $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$.) We now define a function $I : \mathcal{C} \times \{0, 1\}^* \rightarrow \{0, 1\}^*$, so that $I(C, \mathbf{y}) = \langle D_{\mathbf{y}} \rangle$, where $D_{\mathbf{y}}$ has the same number of inputs as C . We condition on whether the input circuit C comes from \mathcal{C}_1 or \mathcal{C}_2 .

Case 1. If $C \in \mathcal{C}_1$, then it has the form $\wedge_m \circ C'$, where C' is a depth- $(d-1)$ circuit with n inputs and m outputs, and \wedge_m is an AND of fan-in m . Let $k \in \mathbb{N}$, and take $n' = (2k/d)^{1/(d-1)}$, so that $f^{d,n'}$ has k inputs. For a k -bit vector \mathbf{y} , we construct $D'_{\mathbf{y}}$ as follows:



Here, $F^{d,n'}$ denotes the obvious depth- d circuit computing the Sipser function $f^{d,n'}$. Now by collapsing the output AND gate of $F^{d,n'}$ into the \wedge_{m+1} , we obtain a depth- d circuit $D_{\mathbf{y}}$ on n inputs such that $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $f^{d,n'}(\mathbf{y}) = 1$, and $D_{\mathbf{y}}(\mathbf{x}) = 0$ otherwise. Hence the restriction of I to $\mathcal{C}_1 \times \{0, 1\}^*$ is an input-switching function for \mathcal{C}_1 and f .

Case 2. If $C \in \mathcal{C}_2$, then it has the form $\vee_m \circ C'$. In this case, we construct the circuit $D'_{\mathbf{y}}$ as follows:



Here $F'^{d,n'}$ denotes the circuit obtained by replacing all AND gates in $F^{d,n'}$ with OR gates and vice-versa, and negating all of the input wires. By collapsing the output OR gate of $F'^{d,n'}$ into the \vee_{m+1} , we obtain a depth- d circuit $D_{\mathbf{y}}$ on n inputs such that $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $f^{d,n'}(\mathbf{y}) = 1$, and $D_{\mathbf{y}}(\mathbf{x}) = 1$ otherwise. Hence the restriction of I to $\mathcal{C}_2 \times \{0, 1\}^*$ is an input-switching function for \mathcal{C}_2 and f .

29:22 Black-Box Hypotheses and Lower Bounds

As before, we observe that each bit in $\langle D_{\mathbf{y}} \rangle$ depends on at most one bit of \mathbf{y} , and that there are circuits C_1 and C_2 which have an AND and OR output gate (respectively) and compute the constant 0 function. Applying Theorem 20, every semantic property P over \mathcal{C} computable in \mathcal{A} is trivial, so the \mathcal{C} -BBH for \mathcal{A} is true. ◀

Geometry of Interaction for ZX-Diagrams

Kostia Chardonnet ✉ 🏠

Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190, Gif-sur-Yvette, France
Université de Paris, CNRS, IRIF, F-75006, Paris, France

Benoît Valiron ✉ 🏠 

Université Paris-Saclay, CNRS, CentraleSupélec, ENS Paris-Saclay, LMF,
91190, Gif-sur-Yvette, France

Renaud Vilmart ✉ 🏠 

Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, LMF, 91190, Gif-sur-Yvette, France

Abstract

ZX-Calculus is a versatile graphical language for quantum computation equipped with an equational theory. Getting inspiration from Geometry of Interaction, in this paper we propose a token-machine-based asynchronous model of both pure ZX-Calculus and its extension to mixed processes. We also show how to connect this new semantics to the usual standard interpretation of ZX-diagrams. This model allows us to have a new look at what ZX-diagrams compute, and give a more local, operational view of the semantics of ZX-diagrams.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Linear logic; Theory of computation → Equational logic and rewriting

Keywords and phrases Quantum Computation, Linear Logic, ZX-Calculus, Geometry of Interaction

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.30

Related Version *Full Version:* <https://hal.archives-ouvertes.fr/hal-03154573/>

Funding This work was supported in part by the French National Research Agency (ANR) under the research project SoftQPRO ANR-17-CE25-0009-02, and by the DGE of the French Ministry of Industry under the research project PIA-GDN/QuantEx P163746-484124.

1 Introduction

Quantum computing is a model of computation where data is stored on the state of particles governed by the laws of quantum physics. The theory is well established enough to have allowed the design of quantum algorithms whose applications are gathering interests from both public and private actors [29, 31, 17].

One of the fundamental properties of quantum objects is to have a *dual* interpretations. In the first one, the quantum object is understood as a *particle*: a definite, localized point in space, distinct from the other particles. Light can be for instance regarded as a set of photons. In the other interpretation, the object is understood as a *wave*: it is “spread-out” in space, possibly featuring interference. This is for instance the interpretation of light as an electromagnetic wave.

The standard model of computation uses *quantum bits* (qubits) for storing information and *quantum circuits* [30] for describing quantum operations with quantum gates, the quantum version of Boolean gates. Although the pervasive model for quantum computation, quantum circuits’ operational semantics is only given in an intuitive manner. A quantum circuit is understood as some sequential, low-level assembly language where quantum gates are opaque black-boxes. In particular, quantum circuits do not natively feature any formal operational semantics giving rise to abstract reasoning, equational theory or well-founded rewrite system.

From a denotational perspective, quantum circuits are literal description of tensors and applications of linear operators. These can be described with the historical matrix interpretation [30], or with the more recent sum-over-paths semantics [1, 6] – this can be



© Kostia Chardonnet, Benoît Valiron, and Renaud Vilmart;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 30; pp. 30:1–30:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

regarded as a *wave-style semantics*. In such a semantics, the state of all of the quantum bits of the memory is mathematically represented as a vector in a (finite dimensional) Hilbert space: the set of quantum bits is a *wave* flowing in the circuit, from the inputs to the output, while the computation generated by the list of quantum gates is a linear map from the Hilbert space of inputs to the Hilbert space of outputs.

In recent years, an alternative model of quantum computation with better formal properties than quantum circuits has emerged: the ZX-Calculus [7]. Originally motivated by a categorical interpretation of quantum theory, the ZX-Calculus is a graphical language that represents linear maps as special kinds of graphs called *diagrams*. Unlike the quantum circuit framework, the ZX-Calculus comes with a sound and complete [24, 33], well-defined equational theory on a small set of canonical generators making it possible to reason on quantum computation by means of local graph rewriting.

The canonical semantics of a ZX diagram consists in a linear operator. This operator can be represented as a matrix or through the more recent sum-over-path semantics [35]. But in both cases, these semantics give a purely functional, *wave-style* interpretation to the diagram. Nonetheless, this graphical language – and its equational theory – has been shown to be amenable to many extensions and is being used in a wide spectrum of applications ranging from quantum circuit optimization [14, 4], verification [25, 15, 13] and representation such as MBQC patterns [16] or error-correction [12, 11].

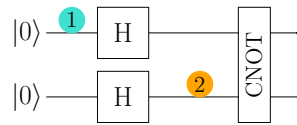
The standard models for both quantum circuits and ZX-Calculus is therefore based on a wave-style interpretation. An alternative operational interpretation of quantum circuit following a *particle-style* semantics has recently been investigated in the literature [9]. In this model, quantum bits are intuitively seen as *tokens* flowing inside the wires of the circuit. Formally, a quantum circuit is interpreted as a token-based automata, based on Geometry of Interaction (GoI) [21, 20, 19, 22]. Among its many instantiations, GoI can be seen as a procedure to interpret a proof-net [23] – graphical representation of proofs of linear logic [18] – as a token-based automaton [10, 2]. The flow of a token inside a proof-net characterises an invariant of the proof – its computational content. This framework is used in [9] to formalize the notion of qubits-as-tokens flowing inside a higher-order term representing a quantum computation – that is, computing a quantum circuit. However, in this work, quantum gates are still regarded as black-boxes, and tokens are purely classical objects *requiring synchronicity*: to fire, a two-qubit gate needs its two arguments to be ready.

As a summary, despite their ad-hoc construction, quantum circuits can be seen from two perspectives: computation as a flow of particles (i.e. tokens), and as a wave passing through the gates. On the other hand, although ZX-Calculus is a well-founded language, it still misses such a particle-style perspective.

In this paper, we aim at giving a novel insight on the computational content of a ZX term in an asynchronous way, emphasizing the graph-like behavior of a ZX-diagram.

Following the idea of using a token machine to exhibit the computational content of a proof-net or a quantum circuit, we present in this paper a token machine for the ZX-Calculus. To exemplify the versatility of the approach, we show how to extend it to mixed processes [8, 5]. To assess the validity of the semantics, we show how it links to the standard interpretation of ZX-diagrams. While the standard interpretation of ZX-diagrams proceeds with diagram decomposition as tensors and products of matrices, the tokens flowing inside the diagram really exploits the connectivity of the diagram.

This ability illustrates one fundamental difference between our approach and the one in [9]. The latter follows a *classical control* approach: if qubits can be in superposition, each qubit inhabits a token sitting in *one single* position in the circuit. For instance, on the circuit on the right, the state of the two tokens



$|\bullet\bullet\rangle$ is $\frac{\sqrt{2}}{2}(|00\rangle + |10\rangle)$. Although the two tokens can be regarded as being in superposition, their *position* is not. In our system, tokens and positions can be superposed. The second fundamental difference lies in the *asynchronicity* of our token-machine. Unlike [9], we rely on the canonical generators of ZX-diagrams: tokens can travel through these nodes in an asynchronous manner. For instance, in the above circuit the orange token has to wait for the blue token before crossing the CNOT gate. As illustrated in Table 1, in our system one token can interact with multi-wire nodes. Finally, as formalized in Theorem 27, a third difference is that compared to [9], the token-machine we present is *non-oriented*: in the circuit above, tokens have to start on the left and flow towards the right of the circuit whereas our system is agnostic on where tokens initially “start”.

Plan of the paper. The paper is organized as follows: in Section 2 we present the ZX-Calculus and its standard interpretation into **Qubit**, and its axiomatization.

In Section 3 we present the actual asynchronous token machine and its semantics and show that it is sound and complete with regard to the standard interpretation of ZX-diagrams. Finally, in Section 4 we present an extension of the ZX-Calculus to mixed processes and adapt the token machine to take this extension into account. Proofs are in the appendix.

2 The ZX-Calculus

The ZX-Calculus is a powerful graphical language for reasoning about quantum computation introduced by Bob Coecke and Ross Duncan [7]. A term in this language is a graph – called a *string diagram* – built from a core set of primitives. In the standard interpretation of ZX-Calculus, a string diagram is interpreted as a matrix. The language is equipped with an equational theory preserving the standard interpretation.

2.1 Pure Operators

The so-called *pure* ZX-diagrams are generated from a set of primitives, given on the right: the Identity, Swap, Cup, Cap, Green-spider and H-gate:

$$\left\{ \begin{array}{l} e_0, \quad \text{Cup} \quad e_1, e_0, \quad \text{Cap} \quad e_1, e_0, \quad \text{H-gate} \quad e_1, e_n, e_0, e_1 \\ \text{Identity} \quad e_0, \quad \text{Swap} \quad e_1, e_0, \quad \text{Green-spider} \quad \alpha, \quad \text{H-gate} \quad e_1, e_n, e_0, e_1 \end{array} \right\}_{\substack{n, m \in \mathbb{N} \\ \alpha \in \mathbb{R} \\ e_i, e'_i \in \mathcal{E}}}$$

We shall be using the following labeling convention: wires (edges) are labeled with e_i , taken from an infinite set of labels \mathcal{E} . We take for granted that distinct wires have distinct labels. The real number α attached to the green spiders is called the *angle*. ZX-diagrams are read top-to-bottom: dangling top edges are the *input edges* and dangling edges at the bottom are *output edges*. For instance, Swap has 2 input and 2 output edges, while Cup has 2 input edges and no output edges. We write $\mathcal{E}(D)$ for the set of edge labels in the diagram D , and $\mathcal{I}(D)$ (resp. $\mathcal{O}(D)$) for the list of input edges (resp. output edges) of D . We denote $::$ the concatenation of lists.

30:4 Geometry of Interaction for ZX-Diagrams

ZX-primitives can be composed either sequentially or in parallel:

$$D_2 \circ D_1 := \begin{array}{c} \dots \\ \boxed{D_1} \\ \dots \\ \boxed{D_2} \\ \dots \end{array} \qquad D_1 \otimes D_2 := \begin{array}{cc} \dots & \dots \\ \boxed{D_1} & \boxed{D_2} \\ \dots & \dots \end{array}$$

We write \mathbf{ZX} for the set of all ZX-diagrams. Notice that when composing diagrams with $(_ \circ _)$, we “join” the outputs of the top diagram with the inputs of the bottom diagram. This requires that the two sets of edges have the same cardinality. The junction is then made by relabeling the input edges of the bottom diagram by the output labels of the top diagram.

► **Convention 1.** We define a second spider, red this time, by composition of Green-spiders and H-gates, as shown below.

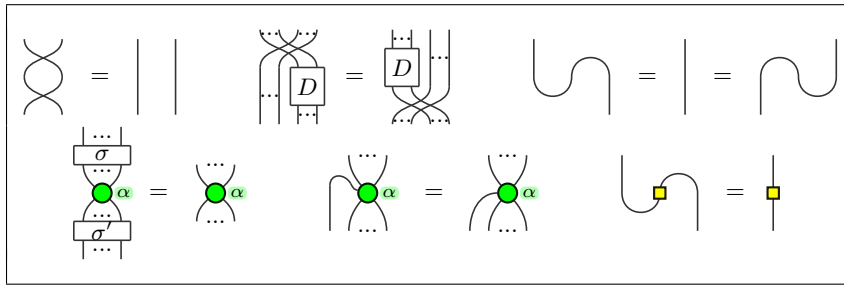
$$\begin{array}{c} \dots \\ \bullet \\ \dots \end{array} := \begin{array}{c} \dots \\ \square \square \\ \bullet \\ \square \square \\ \dots \end{array}$$

► **Convention 2.** We write σ for a permutation of wires, i.e any diagram generated by $\left\{ \begin{array}{c} | \\ | \end{array} , \begin{array}{c} \diagup \\ \diagdown \end{array} \right\}$ with sequential and parallel composition. We write the Cap as η and the Cup as ϵ . We write $Z_k^n(\alpha)$ (resp, X_k^n) for the green-node (resp, red-node) of n inputs, k outputs and parameter α and H for the H-gate. In the remainder of the paper we omit the edge labels when not necessary. Finally, by abuse of notation a green or red node with no explicit parameter holds the angle 0: $\begin{array}{c} \dots \\ \bullet \\ \dots \end{array} := \begin{array}{c} \dots \\ \bullet \\ \dots \end{array}^0$ and $\begin{array}{c} \dots \\ \bullet \\ \dots \end{array} := \begin{array}{c} \dots \\ \bullet \\ \dots \end{array}^0$.

2.2 Standard Interpretation

We understand ZX-diagrams as linear operators through the *standard interpretation*. Informally, wires are interpreted with the two-dimensional Hilbert space, with orthonormal basis written as $\{|0\rangle, |1\rangle\}$, in Dirac notation [30]. Vectors of the form $|.\rangle$ (called “kets”) are considered as columns vector, and therefore $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, and $\alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. Horizontal juxtaposition of wires is interpreted with the *Kronecker*, or *tensor* product. The tensor product of spaces \mathcal{V} and \mathcal{W} whose bases are respectively $\{v_i\}_i$ and $\{w_j\}_j$ is the vector space of basis $\{v_i \otimes w_j\}_{i,j}$, where $v_i \otimes w_j$ is a formal object consisting of a pair of v_i and w_j . We denote $|x\rangle \otimes |y\rangle$ as $|xy\rangle$. In the interpretation of spiders, we use the notation $|0^m\rangle$ to represent an m -fold tensor of $|0\rangle$. As a shortcut notation, we write $|\phi\rangle$ for column vectors consisting of a linear combinations of kets. Shortcut notations are also used for two very useful states: $|+\rangle := \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle := \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Dirac also introduced the notation “bra” $\langle x|$, standing for a row vector. So for instance, $\alpha\langle 0| + \beta\langle 1|$ is $(\alpha \ \beta)$. If $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, we then write $\langle\phi|$ for the vector $\bar{\alpha}\langle 0| + \bar{\beta}\langle 1|$ (with $\bar{(\cdot)}$ the complex conjugation). The notation for tensors of bras is similar to the one for kets. For instance, $\langle x| \otimes \langle y| = \langle xy|$. Using this notation, the scalar product is transparently the product of a row and a column vector: $\langle\phi|\psi\rangle$, and matrices can be written as sums of elements of the form $|\phi\rangle\langle\psi|$. For instance, the identity on \mathbb{C}^2 is $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1|$. For more information on how Hilbert spaces, tensors, compositions and bras and kets work, we invite the reader to consult e.g. [30].

In the standard interpretation [7], a diagram D is mapped to a map between finite dimensional Hilbert spaces of dimensions some powers of 2: $\llbracket D \rrbracket \in \mathbf{Qubit} := \{\mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^m} \mid n, m \in \mathbb{N}\}$.



■ **Figure 1** Connectivity rules. D represents any ZX-diagram, and σ, σ' any permutation of wires.

If D has n inputs and m outputs, its interpretation is a map $\llbracket D \rrbracket : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^m}$ (by abuse of notation we shall use the notation $\llbracket D \rrbracket : n \rightarrow m$). It is defined inductively as follows.

$$\begin{aligned}
 \left[\begin{array}{c} \dots \\ D_1 \\ \dots \\ D_2 \\ \dots \end{array} \right] &= \left[\begin{array}{c} \dots \\ D_2 \\ \dots \end{array} \right] \circ \left[\begin{array}{c} \dots \\ D_1 \\ \dots \end{array} \right] & \left[\begin{array}{c} \dots \\ D_1 \\ \dots \end{array} \right] \left[\begin{array}{c} \dots \\ D_2 \\ \dots \end{array} \right] &= \left[\begin{array}{c} \dots \\ D_1 \\ \dots \end{array} \right] \otimes \left[\begin{array}{c} \dots \\ D_2 \\ \dots \end{array} \right] \\
 \left[\begin{array}{c} | \\ | \end{array} \right] &= id_{\mathbb{C}^2} = |0\rangle\langle 0| + |1\rangle\langle 1| & \left[\begin{array}{c} \diagup \\ \diagdown \end{array} \right] &= \sum_{i,j \in \{0,1\}} |ji\rangle\langle ij| \\
 \left[\begin{array}{c} \curvearrowright \end{array} \right] &= \left[\begin{array}{c} \smile \end{array} \right]^\dagger = |00\rangle + |11\rangle & \left[\begin{array}{c} \blacksquare \end{array} \right] &= |+\rangle\langle 0| + |-\rangle\langle 1| \\
 \left[\begin{array}{c} \dots \\ \alpha \\ \dots \\ \alpha \\ \dots \end{array} \right] &= |0^m\rangle\langle 0^n| + e^{i\alpha} |1^m\rangle\langle 1^n| & \left[\begin{array}{c} \dots \\ \alpha \\ \dots \\ \alpha \\ \dots \end{array} \right] &= |+\rangle\langle +|^n + e^{i\alpha} |-\rangle\langle -|^n
 \end{aligned}$$

2.3 Properties and structure

In this section, we list several definitions and known results that we shall be using in the remainder of the paper. See e.g. [34] for more information.

Universality. ZX-diagrams are *universal* in the sense that for any linear map $f : n \rightarrow m$, there exists a diagram D of **ZX** such that $\llbracket D \rrbracket = f$.

The price to pay for universality is that different diagrams can possibly represent the same quantum operator. There exists however a way to deal with this problem: an equational theory. Several equational theories have been designed for various fragments of the language [3, 26, 24, 27, 28, 33].

Core axiomatization. Despite this variety, any ZX axiomatization builds upon the core set of equations provided in Figure 1, meaning that edges really behave as wires that can be bent, tangled and untangled. They also enforce the irrelevance on the ordering of inputs and outputs for spiders. Most importantly, these rules preserve the standard interpretation given in Section 2.2. We will use these rules – sometimes referred to as “*only connectivity matters*” –, and the fact that they preserve the semantics extensively in the proofs of the results of the paper.

Completeness. The ability to transform a diagram D_1 into a diagram D_2 using the rules of some axiomatization zx (e.g. the core one presented in Figure 1) is denoted $\text{zx} \vdash D_1 = D_2$.

The axiomatization is said to be *complete* whenever any two diagrams representing the same operator can be turned into one another using this axiomatization. Formally:

$$\llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket \iff \text{zx} \vdash D_1 = D_2.$$

It is common in quantum computing to work with restrictions of quantum mechanics. Such restrictions translate to restrictions to particular sets of diagrams – e.g. the $\frac{\pi}{4}$ -fragment which consists of all ZX-diagrams where the angles are multiples of $\frac{\pi}{4}$. There exists axiomatizations that were proven to be complete for the corresponding fragment (all the aforementioned references tackle the problem of completeness).

The developments of this paper are given for the ZX-Calculus in its most general form, but everything in the following also works for fragments of the language.

Input and output wires. An important result which will be used in the rest of the paper is the following:

► **Theorem 3** (Choi-Jamiołkowski). *There are isomorphisms between $\{D \in \mathbf{ZX} \mid D : n \rightarrow m\}$ and $\{D \in \mathbf{ZX} \mid D : n - k \rightarrow k + m\}$ (when $k \leq n$).*

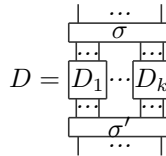
To see how this can be true, simply add cups or caps to turn input edges to output edges (or vice versa), and use the fact that we work modulo the rules of Figure 1.

When $k = n$, this isomorphism is referred to as the *map/state duality*. A related but more obvious isomorphism between ZX-diagrams is obtained by permutation of input wires (resp. output wires).

2.4 Notions of Graph Theory in ZX

Theorem 3 is essential: it allows us to transpose notions of graphs into ZX-Calculus. It is for instance possible to define a notion of connectivity.

► **Definition 4** (Connected Components). *Let D be a non-empty \mathbf{ZX} -diagram. Consider all of the possible decompositions with $D_1, \dots, D_k \in \mathbf{ZX}$ and σ, σ' permutations of wires:*



The largest such k is called the number of connected components of D . It induces a decomposition. The induced D_1, \dots, D_n are called the connected components of D . If D has only one connected component, we say that D is connected.

We can also consider the notions of paths, distance and cycles of usual multi-graphs. We denote $\text{Paths}(e_0, e_n)$ the set of paths from edge e_0 to e_n . We denote $\text{Paths}(D)$ (resp. $\text{Cycles}(D)$) the set of paths (resp. cycles) of diagram D . For a path p , we denote $|p|$ its length. We denote $d(e_0, e_n)$ the distance i.e. the length of the shortest path between e_0 and e_n .

3 A Token Machine for ZX-diagrams

Inspired by the Geometry of Interaction [21, 20, 19, 22] and the associated notion of token machine [10, 2] for proof nets [23], we define here a first token machine on pure ZX-diagrams. The token consists of an edge of the diagram, a direction (either going up, noted \uparrow , or down, noted \downarrow) and a bit (state). The idea is that, starting from an input edge the token will traverse the graph and duplicate itself when encountering an n -ary node (such as the green and red) into each of the input / output edges of the node. Notice that it is not the case for token machines for proof-nets where the token never duplicates itself. This duplication is necessary to make sure we capture the whole linear map encoded by the ZX-diagram. Due

to this duplication, two tokens might collide together when they are on the same edge and going in different directions. The result of such a collision will depend on the states held by both tokens. For a cup, cap or identity diagram, the token will simply traverse it. As for the Hadamard node the token will traverse it and become a superposition of two tokens with opposite states. Therefore, as tokens move through a diagram, some may be added, multiplied together, or annihilated.

► **Definition 5** (Tokens and Token States). *Let D be a ZX-diagram. A token in D is a triplet $(e, d, b) \in \mathcal{E}(D) \times \{\downarrow, \uparrow\} \times \{0, 1\}$. We shall omit the commas and simply write $(e d b)$. The set of tokens on D is written $\mathbf{tk}(D)$. A token state s is then a multivariate polynomial over \mathbb{C} , evaluated in $\mathbf{tk}(D)$. We define $\mathbf{tkS}(D) := \mathbb{C}[\mathbf{tk}(D)]$ the algebra of multivariate polynomials over $\mathbf{tk}(D)$.*

In the token state $t = \sum_i \alpha_i t_{1,i} \cdots t_{n_i,i}$, where the $t_{k,i}$'s are tokens, the components $\alpha_i t_{1,i} \cdots t_{n_i,i}$ are called the terms of t .

A monomial $(e_1 d_1, b_1) \cdots (e_n d_n, b_n)$ encodes the state of n tokens in the process of flowing in the diagram D . A token state is understood as a *superposition* – a linear combination – of multi-tokens flowing in the diagram.

► **Convention 6.** In token states, the sum (+) stands for the superposition while the product stands for additional tokens within a given diagram. We follow the usual convention of algebras of polynomials: for instance, if t_i stands for some token $(e_i d_i b_i)$, then $(t_1 + t_2)t_3 = (t_1 t_3) + (t_2 t_3)$, that is, the superposition of t_1, t_2 flowing in D and t_1, t_3 flowing in D . Similarly, we consider token states modulo commutativity of sum and product, so that for instance the monomial $t_1 t_2$ is the same as $t_2 t_1$. Notice that 0 is an absorbing element for the product ($0 \times t = 0$) and that 1 is a neutral element for the same operation ($1 \times t = t$).

3.1 Diffusion and Collision Rules

The tokens in a ZX-diagram D are meant to move inside D . The set of rules presented in this section describes an *asynchronous* evolution, meaning that given a token state, we will rewrite only one token at a time. The synchronous setting is discussed in Section 5.

► **Definition 7** (Asynchronous Evolution). *Token states on a diagram D are equipped with two transition systems:*

- a collision system (\rightsquigarrow_c), whose effect is to annihilate tokens;
- a diffusion sub-system (\rightsquigarrow_d), defining the flow of tokens within D .

The two systems are defined as follows. With $X \in \{d, c\}$ and $1 \leq j \leq n_i$, if $t_{i,j}$ are tokens in $\mathbf{tk}(D)$, then using Convention 6,






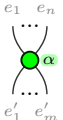




$$\sum_i \alpha_i t_{i,1} \cdots t_{i,j} \cdots t_{i,n_i} \rightsquigarrow_X \sum_i \alpha_i t_{i,1} \cdots \left(\sum_k \beta_k t'_k \right) \cdots t_{i,n_i}$$



provided that $t_{i,j} \rightsquigarrow_X \sum_k \beta_k t'_k$ according to the rules of Table 1. In the table, each rule corresponds to the interaction with the primitive diagram constructor on the left-hand-side. Variables x and b span $\{0, 1\}$, and \neg stands for the negation. In the green-spider rules, $e^{i\alpha x}$ stands for the complex number $\cos(\alpha x) + i \sin(\alpha x)$ and not an edge label.

Finally, as it is customary for rewrite systems, if (\rightarrow) is a step in a transition system, (\rightarrow^) stands for the reflexive, transitive closure of (\rightarrow) .*

We aim at a transition system marrying both collision and diffusion steps. However, for consistency of the system, the order in which we apply them is important as illustrated by the following example.

■ **Table 1** Asynchronous token-state evolution, for all $x, b \in \{0, 1\}$.

	$(e_0 \downarrow x)(e_0 \uparrow x) \rightsquigarrow_c 1$	$(e_0 \downarrow x)(e_0 \uparrow \neg x) \rightsquigarrow_c 0$	(Positive/Negative Collision)
	$(e_b \downarrow x) \rightsquigarrow_d (e_{\neg b} \uparrow x)$		( -diffusion)
	$(e_b \uparrow x) \rightsquigarrow_d (e_{\neg b} \downarrow x)$		( -diffusion)
	$(e_k \downarrow x) \rightsquigarrow_d e^{i\alpha x} \prod_{i \neq k} (e_i \uparrow x) \prod_j (e'_j \downarrow x)$		( -Diffusion)
	$(e'_k \uparrow x) \rightsquigarrow_d e^{i\alpha x} \prod_{j \neq k} (e'_j \downarrow x) \prod_i (e_i \uparrow x)$		
	$(e_0 \downarrow x) \rightsquigarrow_d (-1)^x \frac{1}{\sqrt{2}} (e_1 \downarrow x) + \frac{1}{\sqrt{2}} (e_1 \downarrow \neg x)$		( -Diffusion)
	$(e_1 \uparrow x) \rightsquigarrow_d (-1)^x \frac{1}{\sqrt{2}} (e_0 \uparrow x) + \frac{1}{\sqrt{2}} (e_0 \uparrow \neg x)$		

► **Example 8.** Consider the equality given by the ZX equational theories:  = .

If we drop a token with bit 0 at the top, we hence expect to get a single token with bit 0 at the bottom. We underline the token that is being rewritten at each step. This is what we get when giving the priority to collisions:

$$\begin{array}{c} a \\ \bullet \\ b \text{---} c \\ \bullet \\ d \end{array} \quad :: \quad (a \downarrow 0) \rightsquigarrow_d \underline{(b \downarrow 0)}(c \downarrow 0) \rightsquigarrow_d (d \downarrow 0) \underline{(c \uparrow 0)}(c \downarrow 0) \rightsquigarrow_c (d \downarrow 0)$$

Notice that the collision $(c \uparrow 0)(c \downarrow 0)$ rewrites to 1, and therefore the product $(d \downarrow 0) \times 1 = (d \downarrow 0)$. If however we decide to ignore the priority of collisions, we may end up with a non-terminating run, unable to converge to $(d \downarrow 0)$:

$$(a \downarrow 0) \rightsquigarrow_d \underline{(b \downarrow 0)}(c \downarrow 0) \rightsquigarrow_d (d \downarrow 0) \underline{(c \uparrow 0)}(c \downarrow 0) \rightsquigarrow_d (d \downarrow 0)(a \uparrow 0) \underline{(b \downarrow 0)}(c \downarrow 0) \rightsquigarrow_d \dots$$

We therefore set a rewriting strategy as follows.

► **Definition 9 (Collision-Free).** A token state s of $\mathbf{tkS}(D)$ is called collision-free if for all $s' \in \mathbf{tkS}(D)$, we have $s \not\rightsquigarrow_c s'$.

► **Definition 10 (Token Machine Rewriting System).** We define a transition system \rightsquigarrow as exactly one \rightsquigarrow_d rule followed by all possible \rightsquigarrow_c rules. In other words, $t \rightsquigarrow u$ if and only if there exists t' such that $t \rightsquigarrow_d t' \rightsquigarrow_c^* u$ and u is collision-free.

In [9], a token arriving at an input of a gate is blocked until all the inputs of the gates are populated by a token, at which point all the tokens go through at once (while obviously changing the state). The control is purely classical: it is causal. In our approach, the state of the system is global and there is no explicit notion of qubit. Instead, tokens collect the operation that is to be applied to the input qubits.

3.2 Strong Normalization and Confluence

The token machine Rewrite System of Definition 10 ensures that the collisions that can happen always happen. The system does not a priori forbid two tokens on the same edge, provided that they have the same direction. However this is something we want to avoid as

there is no good intuition behind it: We want to link the token machine to the standard interpretation, which is not possible if two tokens can appear on the same edge.

In this section we show that, under a notion of well-formedness characterizing token uniqueness on each edge, the Token State Rewrite System (\rightsquigarrow) is strongly normalizing and confluent.

► **Definition 11** (Polarity of a Term in a Path). *Let D be a ZX-diagram, and $p \in \text{Paths}(D)$ be a path in D . Let $t = (e, d, x) \in \mathbf{tk}(D)$. Then:*

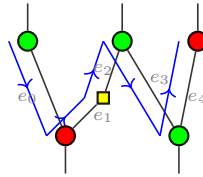
$$P(p, t) = \begin{cases} 1 & \text{if } e \in p \text{ and } e \text{ is } d\text{-oriented} \\ -1 & \text{if } e \in p \text{ and } e \text{ is } \neg d\text{-oriented} \\ 0 & \text{if } e \notin p \end{cases}$$

We extend the definition to subterms $\alpha t_1 \dots t_m$ of a token-state s :

$$P(p, 0) = P(p, 1) = 0, \quad P(p, \alpha t_1 \dots t_m) = P(p, t_1) + \dots + P(p, t_m).$$

In the following, we shall simply refer to such subterms as “terms of s ”.

► **Example 12.** In the (piece of) diagram presented on the right, the blue directed line $p = (e_0, e_1, e_2, e_3, e_4)$ is a path. The orientation of the edges in the path is represented by the arrow heads, and e_3 for instance is \downarrow -oriented in p which implies that we have $P(p, (e_3 \uparrow x)) = -1$.



► **Definition 13** (Well-formedness). *Let D be a ZX-diagram, and $s \in \mathbf{tkS}(D)$ a token state on D . We say that s is well-formed if for every term t in s and every path $p \in \text{Paths}(D)$ we have $P(p, t) \in \{-1, 0, 1\}$.*

► **Proposition 14** (Invariance of Well-Formedness). *Well-formedness is preserved by (\rightsquigarrow): if $s \rightsquigarrow^* s'$ and s is well-formed, then s' is well-formed.*

Well-formedness prevents the unwanted scenario of having two tokens on the same wire, and oriented in the same direction (e.g. $(e_0 \downarrow x)(e_0 \downarrow y)$). As shown in the Proposition 15, this property is in fact stronger.

► **Proposition 15** (Full Characterisation of Well-Formed Terms). *Let D be a ZX-diagram, and $s \in \mathbf{tkS}(D)$ be not well-formed, i.e. there exists a term t in s , and $p \in \text{Paths}(D)$ such that $|P(p, t)| \geq 2$. Then we can rewrite $s \rightsquigarrow s'$ such that a term in s' has a product of at least two tokens of the form $(e_0, d, _)$.*

Although well-formedness prevents products of tokens on the same wire, it does not guarantee termination: for this we need to consider polarities along cycles.

► **Proposition 16** (Invariant on Cycles). *Let D be a ZX-diagram, and $c \in \text{Cycles}(D)$ a cycle. Let t_1, \dots, t_n be tokens, and s be a token state such that $t_1 \dots t_n \rightsquigarrow^* s$. Then for every non-null term t in s we have $P(c, t_1 \dots t_n) = P(c, t)$.*

30:10 Geometry of Interaction for ZX-Diagrams

This proposition tells us that the polarity is preserved inside a cycle. By requiring the polarity to be 0, we can show that the token machine terminates. This property is defined formally in the following.

► **Definition 17** (Cycle-Balanced Token State). *Let D be a ZX-diagram, and t a term in a token state on D . We say that t is cycle-balanced if for all cycles $c \in \text{Cycles}(D)$ we have $P(c, t) = 0$. We say that a token state is cycle-balanced if all its terms are cycle-balanced.*

To show that being cycle-balanced implies termination, we need the following intermediate lemma. This essentially captures the fact that a token in the diagram comes from some other token that “traveled” in the diagram earlier on.

► **Lemma 18** (Rewinding). *Let D be a ZX-diagram, and t be a term in a well-formed token state on D , and such that $t \rightsquigarrow^* \sum_i \lambda_i t_i$, with $(e_n, d, x) \in t_1$. If t is cycle-balanced, then there exists a path $p = (e_0, \dots, e_n) \in \text{Paths}(D)$ such that e_n is d -oriented in p , and $P(p, t) = 1$.*

We can now prove strong-normalization.

► **Theorem 19** (Termination of well-formed, cycle-balanced token state). *Let D be a ZX-diagram, and $s \in \mathbf{tkS}(D)$ be well-formed. The token state s is strongly normalizing if and only if it is cycle-balanced.*

Intuitively, this means that tokens inside a cycle will cancel themselves out if the token state is cycle-balanced. Since cycles are the only way to have a non-terminating token machine, we are sure that our machine will always terminate.

► **Proposition 20** (Local Confluence). *Let D be a ZX-diagram, and $s \in \mathbf{tkS}(D)$ be well-formed and collision-free. Then, for all $s_1, s_2 \in \mathbf{tkS}(D)$ such that $s_1 \leftarrow s \rightsquigarrow s_2$, there exists $s' \in \mathbf{tkS}(D)$ such that $s_1 \rightsquigarrow^* s' \leftarrow s_2$.*

► **Corollary 21** (Confluence). *Let D be a ZX-diagram. The rewrite system \rightsquigarrow is confluent for well-formed, collision-free and cycle-balanced token states.*

► **Corollary 22** (Uniqueness of Normal Forms). *Let D be a ZX-diagram. A well-formed and cycle-balanced token state admits a unique normal form under the rewrite system \rightsquigarrow .*

3.3 Semantics and Structure of Normal Forms

In this section, we discuss the structure of normal forms, and relate the system to the standard interpretation presented in Section 2.

► **Proposition 23** (Single-Token Input). *Let $D : n \rightarrow m$ be a connected ZX-diagram with $\mathcal{I}(D) = [a_i]_{0 < i \leq n}$ and $\mathcal{O}(D) = [b_i]_{0 < i \leq m}$, $0 < k \leq n$ and $x \in \{0, 1\}$, such that:*

$$[[D]] \circ (id_{k-1} \otimes |x\rangle \otimes id_{n-k}) = \sum_{q=1}^{2^{m+n-1}} \lambda_q |y_{1,q}, \dots, y_{m,q}\rangle \langle x_{1,q}, \dots, x_{k-1,q}, x_{k+1,q}, \dots, x_{n,q}|$$

$$\text{Then: } (a_k \downarrow x) \rightsquigarrow^* \sum_{q=1}^{2^{m+n-1}} \lambda_q \prod_i (b_i \downarrow y_{i,q}) \prod_{i \neq k} (a_i \uparrow x_{i,q})$$

This proposition conveys the fact that dropping a single token in state x on wire a_k gives the same semantics as the one obtained from the standard interpretation on the ZX-diagram, with wire a_k connected to the state $|x\rangle$.

Proposition 23 can be made more general. However, we first need the following result on ZX-diagrams:

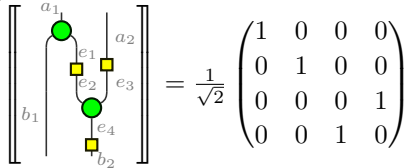
► **Lemma 24** (Universality of Connected ZX-Diagrams). *Let $f : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^m}$. There exists a connected ZX-diagram $D_f : n \rightarrow m$ such that $\llbracket D_f \rrbracket = f$.*

► **Proposition 25** (Multi-Token Input). *Let D be a connected ZX-diagram with $\mathcal{I}(D) = [a_i]_{1 \leq i \leq n}$ and $\mathcal{O}(D) = [b_i]_{1 \leq i \leq m}$; with $n \geq 1$.*

$$\text{If: } \llbracket D \rrbracket \circ \left(\sum_{q=1}^{2^n} \lambda_q |x_{1,q}, \dots, x_{n,q}\rangle \right) = \sum_{q=1}^{2^m} \lambda'_q |y_{1,q}, \dots, y_{m,q}\rangle$$

$$\text{then: } \sum_{q=1}^{2^n} \lambda_q \prod_{i=1}^n (a_i \downarrow x_{i,q}) \rightsquigarrow^* \sum_{q=1}^{2^m} \lambda'_q \prod_{i=1}^m (b_i \downarrow y_{i,q})$$

► **Example 26** (CNOT). In the ZX-Calculus, the CNOT-gate (up to some scalar) can be

constructed as follows:  $= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

On classical inputs, this gate applies the NOT-gate on the second bit if and only if the first bit is at 1. Therefore if we apply the state $|10\rangle$ to it we get $\frac{1}{\sqrt{2}}|11\rangle$.

We demonstrate how the token machine can be used to get this result. Following Proposition 25, we start by initialising the Token Machine in the token state $(a_1 \downarrow 1)(a_2 \downarrow 0)$, matching the input state $|10\rangle$.

We underline each step that is being rewritten, and take the liberty to sometimes do several rewrites in parallel at the same time.

$$\begin{aligned} (a_1 \downarrow 1)(a_2 \downarrow 0) &\rightsquigarrow_d (b_1 \downarrow 1)(e_1 \downarrow 1)(a_2 \downarrow 0) \rightsquigarrow_d (b_1 \downarrow 1)(e_1 \downarrow 1) \frac{1}{\sqrt{2}} \left((e_3 \downarrow 0) + (e_3 \downarrow 1) \right) \\ &\rightsquigarrow_d \frac{1}{\sqrt{2}} (b_1 \downarrow 1)(e_1 \downarrow 1) \left((e_2 \uparrow 0)(e_4 \downarrow 0) + (e_2 \uparrow 1)(e_4 \downarrow 1) \right) \\ &\rightsquigarrow_d \frac{1}{2} (b_1 \downarrow 1) \left((e_2 \downarrow 0) - (e_2 \downarrow 1) \right) \left((e_2 \uparrow 0)(e_4 \downarrow 0) + (e_2 \uparrow 1)(e_4 \downarrow 1) \right) \\ &\rightsquigarrow_c^2 \frac{1}{2} (b_1 \downarrow 1) \left((e_4 \downarrow 0) + ((e_2 \downarrow 0) - (e_2 \downarrow 1))(e_2 \uparrow 1)(e_4 \downarrow 1) \right) \\ &\rightsquigarrow_c^2 \frac{1}{2} (b_1 \downarrow 1) \left((e_4 \downarrow 0) - (e_4 \downarrow 1) \right) \\ &\rightsquigarrow_d \frac{1}{2\sqrt{2}} (b_1 \downarrow 1) \left((b_2 \downarrow 0) + (b_2 \downarrow 1) - (b_2 \downarrow 0) + (b_2 \downarrow 1) \right) \\ &= \frac{1}{\sqrt{2}} (b_1 \downarrow 1)(b_2 \downarrow 1) \end{aligned}$$

The final token state corresponds to $\frac{1}{\sqrt{2}}|11\rangle$, as described by Proposition 25. Notice that during the run, each invariants presented before holds and that due to confluence we could have rewritten the tokens in any order and still obtain the same result.

This proposition is a direct generalization of Proposition 23. It shows we can compute the output of a diagram provided a particular input state. We can also recover the semantics of the whole operator by initialising the starting token state in a particular configuration.

► **Theorem 27** (Arbitrary Wire Initialisation). *Let D be a connected ZX-diagram, with $\mathcal{I}(D) = [a_i]_{1 \leq i \leq n}$, $\mathcal{O}(D) = [b_i]_{1 \leq i \leq m}$, and $e \in \mathcal{E}(D) \neq \emptyset$ such that $(e \downarrow x)(e \uparrow x) \rightsquigarrow^* t_x$ for $x \in \{0, 1\}$ with t_x terminal (the rewriting terminates by Corollary 22). Then:*

$$\llbracket D \rrbracket = \sum_{q=1}^{2^{m+n}} \lambda_q |y_{1,q} \dots y_{m,q}\rangle \langle x_{1,q} \dots x_{n,q}| \implies t_0 + t_1 = \sum_{q=1}^{2^{m+n}} \lambda_q \prod_i (b_i \downarrow y_{i,q}) \prod_i (a_i \uparrow x_{i,q}).$$

30:12 Geometry of Interaction for ZX-Diagrams

► **Example 28.** If we take back the diagram from Example 26 and decide to initialize any wire e of the diagram in the state $(e \downarrow 0)(e \uparrow 0) + (e \downarrow 1)(e \uparrow 1)$ and apply the rewriting as in Theorem 27 we in fact end up with the token state $\frac{1}{\sqrt{2}} \left((a_1 \uparrow 0)(a_2 \uparrow 0)(b_1 \downarrow 0)(b_2 \downarrow 0) + (a_1 \uparrow 0)(a_2 \uparrow 1)(b_1 \downarrow 0)(b_2 \downarrow 1) + (a_1 \uparrow 1)(a_2 \uparrow 0)(b_1 \downarrow 1)(b_2 \downarrow 1) + (a_1 \uparrow 1)(a_2 \uparrow 1)(b_1 \downarrow 1)(b_2 \downarrow 0) \right)$ which matches the actual matrix of the standard interpretation.

► **Remark 29.** At this point, it is legitimate to wonder about the benefits of the token machine over the standard interpretation for computing the semantics of a diagram. Let us first notice that when computing the semantics of a diagram à la Theorem 27, we get in the token state one term per non-null entry in the associated matrix (the one obtained by the standard interpretation).

We can already see that the token-based interpretation can be interesting if the matrix is sparse, the textbook case being Z_n^n whose standard interpretation requires a $2^n \times 2^n$ matrix, while the token-based interpretation only requires two terms (each with $2n$ tokens).

Secondly, we can notice that we can “mimic” the standard interpretation with the token machine. Consider a diagram decomposed as a product of slices (tensor product of generators) for the standard interpretation. Then, for the token machine, without going into technical details, we can follow the strategy that consists in moving token through the diagram one slice at a time. This essentially computes the matrix associated with each slice and its composition.

The point of the token machine however, is that it is versatile enough to allow for more original strategies, some of which may have a worst complexity, but also some of which may have a better one.

4 Extension to Mixed Processes

The token machine presented so far worked for so-called *pure* quantum processes i.e. with no interaction with the environment. To demonstrate how generic our approach is, we show how to adapt it to the natural extension of *mixed* processes, represented with completely positive maps (CPM). This in particular allows us to represent quantum measurements.

4.1 ZX-diagrams for Mixed Processes

The interaction with the environment can be modeled in the ZX-Calculus by adding a unary generator $\underline{\perp}$ to the language [8, 5], that intuitively enforces the state of the wire to be classical. We denote with \mathbf{ZX}^{\perp} the set of diagrams obtained by adding $\underline{\perp}$ this generator.

Similar to what is done in quantum computation, the standard interpretation $\llbracket \cdot \rrbracket^{\perp}$ for \mathbf{ZX}^{\perp} maps diagrams to CPMs. If $D \in \mathbf{ZX}$ we define $\llbracket D \rrbracket^{\perp}$ as $\rho \mapsto \llbracket D \rrbracket^{\dagger} \circ \rho \circ \llbracket D \rrbracket$, and we set $\llbracket \underline{\perp} \rrbracket^{\perp}$ as $\rho \mapsto \text{Tr}(\rho)$, where $\text{Tr}(\rho)$ is the trace of ρ .

There is a canonical way to map a \mathbf{ZX}^{\perp} -diagram to a \mathbf{ZX} -diagram in a way that preserves the semantics: the so-called CPM-construction [32]. We define the map (conveniently named) CPM as the map that preserves compositions $(_ \circ _)$ and $(_ \otimes _)$ and such that:

$$\forall D \in \mathbf{ZX}, \text{CPM} \left(\begin{array}{c} | \dots | \\ \boxed{D} \\ | \dots | \end{array} \right) = \begin{array}{c} \overbrace{\left(\begin{array}{c} \dots \\ \boxed{D} \\ \dots \end{array} \right)}^{\alpha} \\ \underbrace{\left(\begin{array}{c} \dots \\ \boxed{D} \\ \dots \end{array} \right)}_{\alpha} \end{array} \quad \text{CPM}(\underline{\perp}) = \cup$$

Where $\llbracket D \rrbracket^{\text{cj}}$ is D where every angle α has been changed to $-\alpha$.

defined as:
$$\sum_{q=1}^{2^{m+n}} \lambda_q \prod_j (p_j, d_j, x_{j,q}, y_{j,q}) \mapsto \sum_{q=1} \lambda_q \prod_j (p_j, d_j, x_{j,q})(\overline{p_j}, d_j, y_{j,q})$$

Since $\text{CPM}(D)$ can be seen as two copies of D where $\underline{\quad}$ is replaced by \cup , each token in D corresponds to two tokens in $\text{CPM}(D)$, at the same spot but in the two copies of D . The two states x and y represent the states of the two corresponding tokens.

We can then show that this rewriting system is consistent:

► **Theorem 32.** *Let D be a \mathbf{ZX}^{\neq} -diagram, and $t_1, t_2 \in \mathbf{tkS}^{\neq}(D)$. Then whenever $t_1 \rightsquigarrow_{\neq} t_2$ we have $\text{CPM}(t_1) \rightsquigarrow^{\{1,2\}} \text{CPM}(t_2)$.*

The notions of polarity, well-formedness and cycle-balancedness can be adapted, and we get strong normalization (Theorem 19), confluence (Corollary 21), and uniqueness of normal forms (Corollary 22) for well-formed and cycle-balanced token states.

5 Conclusion and Future Work

In this paper, we presented a novel *particle-style* semantics for ZX-Calculus. Based on a token-machine automaton, it emphasizes the *asynchronicity* and *non-orientation* of the computational content of a ZX-diagram. Compared to existing token-based semantics of quantum computation such as [9], our proposal furthermore support decentralized tokens where the position of a token can be in superposition.

As quantum circuits can be mapped to ZX-diagrams, our token machines induce a notion of asynchronicity for quantum circuits. This contrasts with the notion of token machine defined in [9] where some form of synchronicity is enforced.

Our token machines give us a new way to look at how a ZX-diagram computes with a more local, operational approach. This could lead to extensions of the ZX-Calculus with more expressive logical and computational constructs, such as recursion.

As a final remark, we notice that this formalism naturally extends to other graphical languages for qubit quantum computation, and even for tensor networks. It suffices to adapt the diffusion rewriting steps to the generators at hand, which is always possible in the setting of finite dimensional Hilbert spaces, and if needs be to adapt the states in tokens to the dimension of the wire they go through (e.g. if a wire in a tensor network is of dimension 4, the state spans $\{0, 1, 2, 3\}$).

References

- 1 Matthew Amy. Towards Large-scale Functional Verification of Universal Quantum Circuits. In Peter Selinger and Giulio Chiribella, editors, *Proceedings 15th International Conference on Quantum Physics and Logic, QPL 2018, Halifax, Canada, 3-7th June 2018*, volume 287 of *EPTCS*, pages 1–21, 2018. doi:10.4204/EPTCS.287.1.
- 2 Andrea Asperti and Cosimo Laneve. Paths, computations and labels in the λ -calculus. *Theoretical Computer Science*, 142(2):277–297, 1995.
- 3 Miriam Backens. The ZX-Calculus is Complete for Stabilizer Quantum Mechanics. *New Journal of Physics*, 16(9):093021, 2014. doi:10.1088/1367-2630/16/9/093021.
- 4 Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. There and back again: A circuit extraction tale, 2020. arXiv:2003.01664.
- 5 Titouan Carette, Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of Graphical Languages for Mixed States Quantum Mechanics. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International*

- Proceedings in Informatics (LIPIcs)*, pages 108:1–108:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.108.
- 6 Christophe Chareton, Sébastien Bardin, François Bobot, Valentin Perrelle, and Benoît Valiron. A Deductive Verification Framework for Circuit-building Quantum Programs. arXiv:2003.05841. To appear in *Proceedings of ESOP'21*.
 - 7 Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011.
 - 8 Bob Coecke and Simon Perdrix. Environment and Classical Channels in Categorical Quantum Mechanics. *Logical Methods in Computer Science*, Volume 8, Issue 4, November 2012. doi:10.2168/LMCS-8(4:14)2012.
 - 9 Dal Lago, Ugo and Faggian, Claudia and Valiron, Benoît and Yoshimizu, Akira. The geometry of parallelism: Classical, probabilistic, and quantum effects. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, page 833–845, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3009837.3009859.
 - 10 Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal λ -machines. *Theoretical Computer Science*, 227(1-2):79–97, 1999.
 - 11 Niel de Beaudrap, Ross Duncan, Dominic Horsman, and Simon Perdrix. Pauli Fusion: a computational model to realise quantum transformations from ZX terms. In *QPL'19 : International Conference on Quantum Physics and Logic*, Los Angeles, United States, 2019. 12 pages + appendices. URL: <https://hal.archives-ouvertes.fr/hal-02413388>.
 - 12 Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4:218, 2020. doi:10.22331/q-2020-01-09-218.
 - 13 Ross Duncan and Liam Garvie. Verifying the smallest interesting colour code with quantomatic. In Bob Coecke and Aleks Kissinger, editors, *Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, The Netherlands, 3-7 July 2017*, volume 266 of *Electronic Proceedings in Theoretical Computer Science*, pages 147–163, 2018. doi:10.4204/EPTCS.266.10.
 - 14 Ross Duncan, Aleks Kissinger, Simon Perdrix, and John Van De Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, 2020.
 - 15 Ross Duncan and Maxime Lucas. Verifying the Steane code with Quantomatic. In Bob Coecke and Matty Hoban, editors, *Proceedings of the 10th International Workshop on Quantum Physics and Logic, Castelldefels (Barcelona), Spain, 17th to 19th July 2013*, volume 171 of *Electronic Proceedings in Theoretical Computer Science*, pages 33–49. Open Publishing Association, 2014. doi:10.4204/EPTCS.171.4.
 - 16 Ross Duncan and Simon Perdrix. Rewriting Measurement-Based Quantum Computations with Generalised Flow. *Lecture Notes in Computer Science*, 6199:285–296, 2010. doi:10.1007/978-3-642-14162-1_24.
 - 17 Elizabeth Gibney. Quantum gold rush: the private funding pouring into quantum start-ups. *Nature*, 574:22–24, 2019. doi:10.1038/d41586-019-02935-4.
 - 18 Jean-Yves Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.
 - 19 Jean-Yves Girard. Geometry of interaction II: deadlock-free algorithms. In *International Conference on Computer Logic*, pages 76–93. Springer, 1988.
 - 20 Jean-Yves Girard. Geometry of interaction I: interpretation of System F. In *Studies in Logic and the Foundations of Mathematics*, volume 127, pages 221–260. Elsevier, 1989.
 - 21 Jean-Yves Girard. Towards a geometry of interaction. *Contemporary Mathematics*, 92(69-108):6, 1989.
 - 22 Jean-Yves Girard. Geometry of interaction III: accommodating the additives. *London Mathematical Society Lecture Note Series*, pages 329–389, 1995.
 - 23 Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. *Lecture Notes in Pure and Applied Mathematics*, pages 97–124, 1996.

- 24 Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. Two Complete Axiomatisations of Pure-state Qubit Quantum Computing. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18*, pages 502–511, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209128.
- 25 Anne Hillebrand. Quantum Protocols involving Multiparticle Entanglement and their Representations. Master's thesis, University of Oxford, 2011. URL: <https://www.cs.ox.ac.uk/people/bob.coecke/Anne.pdf>.
- 26 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18*, pages 559–568, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209131.
- 27 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Diagrammatic Reasoning Beyond Clifford+T Quantum Mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18*, pages 569–578, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209139.
- 28 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A Generic Normal Form for ZX-Diagrams and Application to the Rational Angle Completeness. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, June 2019. doi:10.1109/LICS.2019.8785754.
- 29 Alexandre Ménard, Ivan Ostojic, Mark Patel, and Daniel Volz. A game plan for quantum computing. *McKinsey Quarterly*, 2020.
- 30 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2002.
- 31 Qureca.com. Overview on quantum initiatives worldwide. <https://www.qureca.com/overview-on-quantum-initiatives-worldwide/>, January 2021.
- 32 Peter Selinger. Dagger compact closed categories and completely positive maps. *Electronic Notes in Theoretical computer science*, 170:139–163, 2007.
- 33 Renaud Vilmart. A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, June 2019. doi:10.1109/LICS.2019.8785765.
- 34 Renaud Vilmart. *ZX-Calculi for Quantum Computing and their Completeness*. Theses, Université de Lorraine, 2019. URL: <https://hal.archives-ouvertes.fr/tel-02395443>.
- 35 Renaud Vilmart. The Structure of Sum-Over-Paths, its Consequences, and Completeness for Clifford, 2020. arXiv:2003.05678.

Diameter Versus Certificate Complexity of Boolean Functions

Siddhesh Chaubal ✉

University of Texas at Austin, TX, USA

Anna Gál ✉

University of Texas at Austin, TX, USA

Abstract

In this paper, we introduce a measure of Boolean functions we call *diameter*, that captures the relationship between certificate complexity and several other measures of Boolean functions. Our measure can be viewed as a variation on alternating number, but while alternating number can be exponentially larger than certificate complexity, we show that diameter is always upper bounded by certificate complexity. We argue that estimating diameter may help to get improved bounds on certificate complexity in terms of sensitivity, and other measures.

Previous results due to Lin and Zhang [20] imply that $s(f) \geq \Omega(n^{1/3})$ for transitive functions with constant alternating number. We improve and extend this bound and prove that $s(f) \geq \sqrt{n}$ for transitive functions with constant alternating number, as well as for transitive functions with constant diameter. We also show that $bs(f) \geq \Omega(n^{3/7})$ for transitive functions under the weaker condition that the “minimum” diameter is constant.

Furthermore, we prove that the log-rank conjecture holds for functions of the form $f(x \oplus y)$ for functions f with diameter bounded above by a polynomial of the logarithm of the Fourier sparsity of the function f .

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Sensitivity Conjecture, Boolean Functions, Certificate Complexity, Block Sensitivity, Log-rank Conjecture, Alternating Number

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.31

Acknowledgements We thank the anonymous referees for helpful comments on a previous version of the paper.

1 Introduction

The alternating number of a Boolean function f , denoted $alt(f)$, measures how close the function is to being monotone. It was first studied by Markov [22], who showed that the minimum number of negation gates to compute f by any Boolean circuit is exactly $\lceil \log_2(alt(f) + 1) \rceil$. This led to further studies of alternating number in connection to understanding the effect of negation gates in various contexts such as circuit complexity [31, 34, 26, 27], learning theory [8], and cryptography [16].

Our work is motivated by an interesting paper of Lin and Zhang [20], who studied functions with small alternating number in the context of the sensitivity conjecture, and the log-rank conjecture for XOR functions. The sensitivity conjecture of Nisan and Szegedy [28] states that several important complexity measures, for example block sensitivity $bs(f)$, certificate complexity $C(f)$, and degree $deg(f)$ (over the reals) are all upper bounded by a polynomial of sensitivity $s(f)$. The sensitivity conjecture has been recently proved by Huang [17], who showed that for any Boolean function f , $deg(f) \leq s(f)^2$. Huang’s result was further strengthened by Laplante et al. [19] and Aaronson et al. [1]. Both conjectures have been open for several decades, and – until Huang’s result resolving the sensitivity conjecture – were verified only for a few special classes of Boolean functions. The log-rank conjecture is still open even for XOR functions.



© Siddhesh Chaubal and Anna Gál;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 31; pp. 31:1–31:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

31:2 Diameter vs. Certificate Complexity

Both conjectures can be easily verified to hold for monotone Boolean functions. Lin and Zhang [20] showed that both conjectures remain true for functions that are close to monotone, that is for functions with small alternating number. More precisely, they showed that the conjectures hold for functions with constant alternating number, as well as for functions with alternating number bounded above by some relevant complexity measures of the functions, such as sensitivity in the case of the sensitivity conjecture, and Fourier sparsity in the case of the log-rank conjecture for XOR functions. Thus, their work extended the class of functions where the conjectures can be verified. On the other hand, Dinesh and Sarma [14] presented a function f such that $alt(f)$ is exponentially larger than the certificate complexity $C(f)$. This means that the sensitivity conjecture and the log-rank conjecture for XOR functions cannot be proved in the general case by providing upper bounds on alternating number.

Diameter of Boolean functions. In this paper, we introduce a measure of Boolean functions we call *diameter*, that captures the relationship between certificate complexity and several other measures of Boolean functions. To define the diameter of a Boolean function f , we first consider the “distance” between a vertex of the Boolean cube and subcubes where the function f is constant. However, instead of measuring distance by the number of edges of the Boolean cube along a path (which would correspond to Hamming distance), we allow flipping several bits in one step. We consider paths in the Boolean cube, where one step of the path involves flipping a block of input bits of the function, thus each step specifies a subcube. We require that each step along a path corresponds to a subcube, where the subfunction of f satisfies certain conditions. We define several variants, by considering different classes of Boolean functions that can appear as subfunctions on the subcubes associated with the steps of a path.

For example, in the first variant of our measure, the requirement is that each step along the path corresponds to flipping bits of a minimal sensitive block of the function f . That is, for a step $x^{(i)}, x^{(i+1)}$ along such a path, the requirement is that $f(x^{(i)}) \neq f(x^{(i+1)})$, but $f(x^{(i)}) = f(y)$ for every $y \neq x^{(i+1)}$ from the subcube defined by the bits where $x^{(i)}$ and $x^{(i+1)}$ differ. Notice that this condition means that the subfunction of f restricted to the subcube associated with each step is either the AND function (when $f(x^{(i)}) = 0$) or the NAND function (when $f(x^{(i)}) = 1$). Generalizing this idea, we define several variants of our measure by specifying the class \mathcal{H} of functions that can appear as subfunctions along the steps of a legal path.

Once we specified which steps are legal along a path for a given variant of our measure, we proceed as follows. For a given input x , we define the diameter of f on x as the length of the shortest “legal” path from \bar{x} (the complement of x) to any certificate of f on x , that is to any constant subcube containing x . Then, similarly to standard complexity measures like certificate complexity, we take the maximum over all inputs. Depending on the class \mathcal{H} of functions we allow to appear as subfunctions along the steps of a path, we obtain variants of our measure.

Comparison of diameter with alternating number. Our measure is motivated by alternating number, but it is quite different from it.

First, the similarity is that both measures involve considering paths in the Boolean cube, where one step of the path involves flipping a block of input bits of the function, thus each step specifies a subcube. For alternating number, the requirement on the function values on the subcubes is that the function takes different values on the two opposite (all 0 and all 1) points of the subcubes. For our measure, we consider different classes of Boolean functions that can appear as subfunctions on the subcubes associated with a path.

We note that the definition of alternating number also requires that a path is monotone, (that is the set of 1 bits of an input x on the path must be a subset of the 1 bits of any input that appears later in the path). We do not impose such requirement. In contrast to alternating number, diameter does not measure closeness to monotonicity.

Furthermore, alternating number considers the longest “legal” path between just two specific points, the all 0 input and the all 1 input. For our measure, we consider the shortest “legal” path between inputs x and subcubes corresponding to certificates of the function on the complementary input \bar{x} .

We note that in general, diameter and alternating number are incomparable, and we provide examples that illustrate this. However, an important distinction is that while alternating number can be exponentially larger than certificate complexity [14], each variant of our measure considered in this paper is upper bounded by certificate complexity, up to constant factors.

Diameter vs certificate complexity. We define the following variants, depending on the class \mathcal{H} of functions we allow to appear as subfunctions along the steps of a path: dia_{\wedge} where \mathcal{H} consists of the functions AND and NAND (these are the possible subfunctions associated with minimal sensitive blocks, as discussed above), dia_s where \mathcal{H} includes all functions with full sensitivity, dia_{deg} where \mathcal{H} includes all functions with full real degree, dia_{deg_2} where \mathcal{H} includes all functions with full \mathbb{F}_2 -degree.

Note that each of the classes we consider contains the functions AND and NAND since both of these functions have full sensitivity, full \mathbb{F}_2 -degree and full real degree. Thus, each of the measures $dia_s(f)$, $dia_{deg}(f)$ and $dia_{deg_2}(f)$ is upper bounded by $dia_{\wedge}(f)$, for every Boolean function f . On the other hand, as we illustrate by examples, $dia_s(f)$, $dia_{deg}(f)$ and $dia_{deg_2}(f)$ may be significantly smaller than $dia_{\wedge}(f)$, thus considering these variants may lead to stronger bounds. Furthermore, since $deg_2(f) \leq deg(f)$, we have that $dia_{deg}(f) \leq dia_{deg_2}(f)$. We also present examples showing that $dia_{deg}(f)$ may be significantly smaller than $dia_{deg_2}(f)$.

We prove that for all the classes \mathcal{H} considered in this paper,

$$dia_{\mathcal{H}}(f) \leq dia_{\wedge}(f) \leq 2C(f).$$

Depending on the class \mathcal{H} , we can lower bound $dia_{\mathcal{H}}$ by certificate complexity divided by specific complexity measures, such as sensitivity. We show that for any Boolean function f , $C(f)/s(f)$ is upper bounded by $dia_s(f)$ and thus we can upper bound certificate complexity as follows:

$$C(f) \leq dia_s(f)s(f) \leq dia_{\wedge}(f)s(f).$$

Similarly, considering the classes \mathcal{H} consisting of Boolean functions with full real degree and full \mathbb{F}_2 -degree, respectively, we get the following bounds relating diameter and certificate complexity. For any Boolean function f , $C(f) \leq dia_{deg}(f)deg(f) \leq dia_{\wedge}(f)deg(f)$ and $C(f) \leq dia_{deg_2}(f)deg_2(f) \leq dia_{\wedge}(f)deg_2(f)$.

Other variants. One could consider more versions of our measure, for various other classes \mathcal{H} . Another class that is natural to consider in connection to the log-rank conjecture for XOR functions is taking \mathcal{H} to be the class of Boolean functions with full Fourier sparsity, that is functions such that all their Fourier coefficients are nonzero. We do not discuss this variant in more details, as the results are analogous to our results on dia_{deg_2} with similar applications.

31:4 Diameter vs. Certificate Complexity

We would like to mention another version of our definitions, that turns out to be helpful in proving some of our results for special classes of Boolean functions. Several papers in the literature consider minimum certificate complexity, defined as $C_{min}(f) = \min_x C(f, x)$. Similarly, while we define diameter as $dia_{\mathcal{H}}(f) = \max_x dia_{\mathcal{H}}(f, x)$, we also consider minimum diameter defined as $dia_{min, \mathcal{H}}(f) = \min_x dia_{\mathcal{H}}(f, x)$.

Results on transitive functions with small diameter. There has been a long line of work trying to estimate the sensitivity and block sensitivity for transitive functions [32] and also for special classes of transitive functions such as symmetric functions and graph properties [38, 33], minterm transitive functions and cyclically invariant functions [10, 2, 15], transitive functions with sparse DNFs [12].

It has been conjectured that all transitive functions must have “large” sensitivity and block sensitivity. No examples of transitive functions are known on n input bits with $o(n^{1/3})$ sensitivity. Chakraborty [10] constructed a transitive function on n variables with sensitivity $\Theta(n^{1/3})$. It is noted in [12] that an argument in [32] together with Huang’s result gives that any transitive function f on n variables has $s(f) \geq \Omega(n^{1/6})$.

Previous results due to Lin and Zhang [20] imply that $s(f) \geq \Omega(n^{1/3})$ for transitive functions with constant alternating number. We improve and extend this bound and prove that $s(f) \geq \sqrt{n}$ for transitive functions with constant alternating number, as well as for transitive functions with constant diameter, considering dia_s or dia_{\wedge} .

Regarding block sensitivity, it has been conjectured that $\Omega(n^{3/7})$ is a lower bound on the block sensitivity of all transitive functions. There is an example of a transitive function f due to Amano [2] that has $bs(f) = \theta(n^{3/7})$, and no transitive function is known with smaller block sensitivity. Sun [32] proved a lower bound of $n^{1/3}$ on the block sensitivity for all transitive functions. Since block sensitivity is at least sensitivity, our result above also implies that for transitive functions with constant alternating number or constant diameter (dia_s or dia_{\wedge}), $bs(f) \geq \Omega(\sqrt{n})$. We prove that the conjectured $\Omega(n^{3/7})$ lower bound holds under a weaker condition, for all transitive functions with constant *minimum* diameter ($dia_{min, s}$ or $dia_{min, \wedge}$).

Log-rank conjecture for XOR functions with small diameter. The log-rank conjecture for functions of the form $f(x \oplus y)$ has been proved when f belongs to certain special classes such as monotone or linear threshold functions [25], symmetric functions [39], functions with low \mathbb{F}_2 -degree or small spectral norm [35], AC^0 functions [18], read- k functions [11], and functions with constant alternating number by Lin and Zhang [20]. We prove that the log-rank conjecture holds for functions of the form $f(x \oplus y)$ for functions f with dia_{deg_2} or dia_{\wedge} bounded above by a polynomial of the logarithm of the Fourier sparsity of f .

Further motivation for considering diameter. As we noted above, $dia_{\mathcal{H}}(f) \leq 2C(f)$ for any f and any class \mathcal{H} that contains the functions AND and NAND. Huang’s result implies that $C(f) = O(s(f)^5)$, which in turn implies that for any f , and any class \mathcal{H} that contains the functions AND and NAND, (which means for every class considered in this paper), we have

$$dia_{\mathcal{H}}(f) = O(s(f)^5).$$

Obtaining new upper bounds on $dia_{\mathcal{H}}$ could lead to the following interesting consequences:

- An independent proof of the upper bound $dia_{\mathcal{H}}(f) \leq poly(s(f))$ for dia_s or dia_{\wedge} could lead to an independent, purely combinatorial proof of the sensitivity conjecture.

- Improving the upper bound on dia_s or dia_\wedge in terms of sensitivity to $dia_{\mathcal{H}}(f) \leq O(s(f)^2)$, would improve the current best upper bounds on block sensitivity and certificate complexity to $bs(f) \leq C(f) \leq O(s(f)^3)$.

In connection to this question, we note that there are Boolean functions with $dia_s(f)$ (and thus $dia_\wedge(f)$) at least $\Omega(s(f)^{2-o(1)})$, since [3] exhibited a function with $C(f) = \Omega(s(f)^{3-o(1)})$ improving previous results of [6] and [5].

- Proving that $dia_{deg}(f) \leq deg(f)$ would imply the bound $C(f) \leq O(s(f)^4)$, using Huang's result, but improving its current implication which gives only $C(f) \leq O(s(f)^5)$. It would also imply that $C(f) \leq O(deg(f)^2)$, improving the current best bound giving $C(f) \leq deg(f)^3$ by [24].

We note that there are Boolean functions with $dia_{deg}(f)$ (and thus $dia_{deg_2}(f)$ and $dia_\wedge(f)$) at least $\Omega(deg(f)^{1-o(1)})$, since there are Boolean functions with $C(f) = \Omega(deg(f)^{2-o(1)})$ which was shown recently in [3] improving the previous $\Omega(deg(f)^{1.63})$ bound of [29].

- Upper bounds on $dia_{\mathcal{H}}$ (considering an appropriate \mathcal{H}) for specific classes of Boolean functions could give stronger upper bounds on block sensitivity or certificate complexity in terms of sensitivity than currently known for these classes, and could verify the log-rank conjecture for XOR functions for new classes.

2 Preliminaries

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function and $x \in \{0, 1\}^n$ be any input. For $i \in [n]$ we denote by x^i the input obtained by flipping the i -th bit of x . More generally, for $S \subseteq [n]$ we denote by x^S the input obtained by flipping the bits of x in all coordinates in the subset S .

For any two inputs $x, y \in \{0, 1\}^n$, we say $x \prec y$ if $x_i \leq y_i$ for all $i \in [n]$.

► **Definition 2.1** (Sensitivity). *The sensitivity $s(f, x)$ of a Boolean function f on input x is the number of coordinates $i \in [n]$ such that $f(x) \neq f(x^i)$. The sensitivity of f is defined as $s(f) = \max\{s(f, x) : x \in \{0, 1\}^n\}$.*

► **Definition 2.2** (Block Sensitivity). *The block sensitivity $bs(f, x)$ of a Boolean function f on input x is the maximum number of pairwise disjoint subsets S_1, \dots, S_k of $[n]$ such that for each $i \in [k]$ $f(x) \neq f(x^{S_i})$. The block sensitivity of f is defined as $bs(f) = \max\{bs(f, x) : x \in \{0, 1\}^n\}$.*

► **Definition 2.3** (Partial assignment, subcube and subfunction). *Given an integer $n > 0$, a partial assignment α is a function $\alpha : [n] \rightarrow \{0, 1, \star\}$. A partial assignment α corresponds naturally to a setting of n variables (x_1, x_2, \dots, x_n) to $\{0, 1, \star\}$ where x_i is set to $\alpha(i)$.*

The variables set to \star are called unassigned or free, and we say that the variables set to 0 or 1 are fixed. We say that $x \in \{0, 1\}^n$ agrees with α if $x_i = \alpha(i)$ for all i such that $\alpha(i) \neq \star$. The set of all inputs $x \in \{0, 1\}^n$ agreeing with α constitutes a subcube which we denote by \mathcal{S}_α .

The size of a partial assignment α is defined as the number of fixed variables of α and denoted as $|\alpha|$.

For a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, we denote by f_α the subfunction obtained by restricting f to the subcube \mathcal{S}_α .

► **Definition 2.4** (Certificate). *For a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and input $x \in \{0, 1\}^n$ a partial assignment α is a certificate of f on x if x agrees with α , and any input y agreeing with α satisfies $f(y) = f(x)$.*

We denote the set of all certificates of f on x by $\Gamma_f(x)$.

31:6 Diameter vs. Certificate Complexity

► **Definition 2.5** (Certificate Complexity). *The certificate complexity $C(f, x)$ of a Boolean function f on input x is the size of the smallest certificate of f on x . The certificate complexity of f is defined as $C(f) = \max\{C(f, x) : x \in \{0, 1\}^n\}$. The minimum certificate complexity of f is defined as $C_{\min}(f) = \min\{C(f, x) : x \in \{0, 1\}^n\}$.*

► **Definition 2.6** (Alternating path). *For a Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, an alternating path is defined as any sequence of inputs $x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(t)}$, $x^{(i)} \in \{0, 1\}^n$ for $i \in \{0, 1, \dots, t\}$, that satisfies the following properties:*

- $x^{(0)} = 0^n$
- $x^{(i)} \prec x^{(i+1)}$ for all $i \in \{0, 1, \dots, t-1\}$.
- $f(x^{(i)}) \neq f(x^{(i+1)})$ for all $i \in \{0, 1, \dots, t-1\}$

where $a \prec b$ denotes the property that the set of bits set to 1 in a forms a subset of the set of bits set to 1 in b .

► **Definition 2.7** (Alternating Number of a function). *For a Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, the alternating number of f , $\text{alt}(f)$, is defined as the maximum length of any alternating path of f .*

► **Definition 2.8** (Invariance Group). *A Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ is invariant under a permutation $\sigma: [n] \rightarrow [n]$, if for any $x \in \{0, 1\}^n$, $f(x_1, \dots, x_n) = f(x_{\sigma(1)}, \dots, x_{\sigma(n)})$. The set of all permutations under which f is invariant forms a group, called the invariance group of f .*

► **Definition 2.9** (Transitive Function). *A Boolean function is transitive if its invariance group Γ is transitive, that is, for each $i, j \in [n]$, there is a $\sigma \in \Gamma$ such that $\sigma(i) = j$.*

For example, the set of all permutations on n bits, denoted by S_n is a transitive group of permutations. Another example of a transitive group of permutations is the set of all *cyclic shifts* on n bits, denoted by $\text{Shift}_n = \{\xi_0, \xi_1, \dots, \xi_{n-1}\}$, where the permutation ξ_j cyclically shifts the string by j positions.

Communication Complexity. We consider a setting with two parties Alice and Bob and a fixed Boolean function $f: \{0, 1\}^{2n} \rightarrow \{0, 1\}$. For $x, y \in \{0, 1\}^n$, input x is provided to Alice and input y to Bob. Their collective objective is to compute $f(x, y)$.

The communication complexity of f , denoted $CC(f)$, is the maximum value of the minimum number of bits exchanged by Alice and Bob in order to compute $f(x, y)$, where the maximum is taken over all input pairs $(x, y) \in \{0, 1\}^{2n}$.

The communication matrix corresponding to f , denoted M_f , is a $2^n \times 2^n$ matrix with rows indexed by all possible values of $x \in \{0, 1\}^n$ i.e. Alice's part of the input and columns indexed by all possible values of $y \in \{0, 1\}^n$ i.e. Bob's part of the input. It can be shown that $CC(f) \geq \log \text{rank}(M_f)$ [23].

The Log-rank conjecture proposed by Lovász and Saks [21] asks whether the communication complexity of a function can also be upper bounded by a polynomial in logarithm of the rank of its communication matrix as:

► **Conjecture 2.10.** *For any function $f: \{0, 1\}^{2n} \rightarrow \{0, 1\}$,*

$$CC(f) \leq \text{poly}(\log \text{rank}(M_f))$$

Please see Section A in the Appendix for more definitions and background.

2.1 Previous results

We now state some previous results that we use in our proofs.

First we state a couple of lemmas from a recent paper by Chabul and Gál [12].

► **Lemma 2.11** (Lemma 8 from [12]). *For any non-constant transitive function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, we have $C(f, 0^n) \cdot s(f) \geq n$ and $C(f, 1^n) \cdot s(f) \geq n$.*

► **Lemma 2.12** (Lemma 9 from [12]). *For any non-constant transitive function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, and an integer $5 \leq r \leq 15$, if $C_{\min}(f) \leq n^{3/r}$, then $bs(f) \geq \Omega(n^{1-\frac{4}{r}})$.*

We now state a lemma from the paper of Lin and Zhang [20]:

► **Lemma 2.13** (Lemma 12 from [20]). *For any function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, the following two statements hold:*

$$\max\{C(f, 0^n), C(f, 1^n)\} \leq alt(f) \cdot s(f)$$

$$\max\{C(f, 0^n), C(f, 1^n)\} \leq alt(f) \cdot deg_2(f)$$

We include more details about the approach and previous results of Lin and Zhang [20] in Section A.1 in the Appendix.

3 Diameters of Boolean functions

We begin with some notation. For the Boolean cube \mathcal{B}_n , a path is any sequence of inputs $x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(t)}$ where $x^{(i)} \in \{0, 1\}^n$ for $i \in \{0, 1, \dots, t\}$. We define the length of such a path to be the number of steps t . For a path $x^{(0)}, x^{(1)}, x^{(2)}, \dots, x^{(t)}$, we define a sequence of partial assignments $\{\beta^{(0)}, \beta^{(1)}, \dots, \beta^{(t-1)}\}$ where $\beta^{(i)} : [n] \rightarrow \{0, 1, \star\}$ is defined as follows: $\beta_j^{(i)} = x_j^{(i)}$ for all $j \in [n]$ such that $x_j^{(i)} = x_j^{(i+1)}$ and $\beta_j^{(i)} = \star$ otherwise.

Note that we can view each step $x^{(i)} \rightarrow x^{(i+1)}$ on a path as flipping the bits where $x^{(i)}$ and $x^{(i+1)}$ differ. The free variables of the partial assignment $\beta^{(i)}$ are exactly these bits. Thus, for a Boolean function f , the subfunction $f_{\beta^{(i)}}$ depends on the bits where $x^{(i)}$ and $x^{(i+1)}$ differ.

► **Definition 3.1** (\mathcal{H} -distance between an input and a certificate). *Let \mathcal{H} be a class of Boolean functions. For a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, input $x \in \{0, 1\}^n$ and a partial assignment $\alpha : [n] \rightarrow \{0, 1, \star\}$ corresponding to a subcube where f is constant, we define an \mathcal{H} -path from x to α as any path $x^{(0)}, x^{(1)}, \dots, x^{(t)}$ that satisfies the following properties:*

- $x^{(0)} = x$
- $x^{(t)}$ agrees with α
- The subfunction $f_{\beta^{(i)}}$ belongs to class \mathcal{H} for each $i \in \{0, 1, \dots, t-1\}$.

We define the \mathcal{H} -distance between x and α with respect to f , denoted $dist_{f, \mathcal{H}}(x, \alpha)$, to be the length of a shortest \mathcal{H} -path from x to α .

► **Definition 3.2** (\mathcal{H} -diameter of a function). *For a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, input $x \in \{0, 1\}^n$, and a class of Boolean functions \mathcal{H} , we use the notation*

$$dia_{\mathcal{H}}(f, x) = \min_{\alpha \in \Gamma_f(x)} dist_{f, \mathcal{H}}(\bar{x}, \alpha).$$

Recall that \bar{x} denotes the complement of x and $\Gamma_f(x)$ denotes the set of all certificates of f on x .

We define the \mathcal{H} -diameter of f as:

$$dia_{\mathcal{H}}(f) = \max_{x \in \{0, 1\}^n} dia_{\mathcal{H}}(f, x).$$

31:8 Diameter vs. Certificate Complexity

In this work, we will be concerned with the \mathcal{H} -diameter of functions for the following classes \mathcal{H} :

- AND diameter denoted $dia_{\wedge}(f)$: Corresponds to the class \mathcal{H} that includes the function AND and its negation the NAND function.
- Sensitivity diameter denoted $dia_s(f)$: Defined by the class \mathcal{H} with functions that have sensitivity equal to the number of input variables.
- Real degree diameter denoted as $dia_{deg}(f)$: Corresponding to the class of functions \mathcal{H} with real degree equal to the number of input variables.
- \mathbb{F}_2 -degree diameter denoted as $dia_{deg_2}(f)$: Defined by the class \mathcal{H} that include functions with \mathbb{F}_2 -degree equal to the number of variables.

Note that since the functions AND and NAND belong to each of the classes we consider in this paper, and since $deg_2(f) \leq deg(f)$ for any Boolean function f , we have

$$dia_s(f) \leq dia_{\wedge}(f) \tag{1}$$

and

$$dia_{deg}(f) \leq dia_{deg_2}(f) \leq dia_{\wedge}(f) \tag{2}$$

Similarly to minimum certificate complexity, we also define the minimum version of the diameter:

► **Definition 3.3** (Min \mathcal{H} -diameter of a function). *For a Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ and a class of Boolean functions \mathcal{H} , we define the min \mathcal{H} -diameter of f as:*

$$dia_{min, \mathcal{H}}(f) = \min_{x \in \{0, 1\}^n} dia_{\mathcal{H}}(f, x).$$

and we define the closure of the min \mathcal{H} -diameter of f as:

$$dia_{min, \mathcal{H}}^{clo}(f) = \max_{\alpha} dia_{min, \mathcal{H}}(f_{\alpha})$$

where the maximum is taken over all possible partial assignments α on n variables, or in other words, over all possible subfunctions f_{α} of f .

Note that for any function f and class \mathcal{H} , we have $dia_{min, \mathcal{H}}(f) \leq dia_{min, \mathcal{H}}^{clo}(f) \leq dia_{\mathcal{H}}(f)$.

3.1 Upper bounds on diameters

► **Lemma 3.4.** *For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we have:*

$$dia_{\wedge}(f) \leq 2C(f)$$

Proof. For any input $x \in \{0, 1\}^n$, we shall prove that $dia_{\wedge}(f, x) \leq 2C(f, x)$.

Let α be a certificate of f on x achieving $|\alpha| = C(f, x)$. We shall construct an AND-NAND path $x^{(0)}, x^{(1)}, \dots, x^{(t)}$ from \bar{x} to α , with length $t \leq 2|\alpha|$.

We start with $x^{(0)} = \bar{x}$ as required. We now give an inductive description of our construction. Let us assume that we have found the first $i + 1$ vertices of the path i.e. $x^{(0)}, x^{(1)}, \dots, x^{(i)}$. Then we get the next vertex $x^{(i+1)}$ in the following way based on the value of $f(x^{(i)})$:

1. Case 1: $f(x^{(i)}) = f(x)$

If $x^{(i)}$ agrees with α , then we have successfully found the required \mathcal{H} -path from \bar{x} to α and we can stop.

Otherwise, if $x^{(i)}$ does not agree with α , then we choose $x^{(i+1)} = (x^{(i)})^S$ where S is any minimal sensitive block of f on $x^{(i)}$ such that S does not contain any index where $x^{(i)}$ and α agree. Note that such a block S must exist because otherwise, the set of bits where $x^{(i)}$ and α agree would be a certificate of f , which would contradict the minimality of the certificate α .

We note that in this case, $x^{(i+1)}$ agrees with α on at least as many bits as $x^{(i)}$ agrees with α .

2. Case 2: $f(x^{(i)}) \neq f(x)$

In this case, we first observe that the set T of all the bits where $x^{(i)}$ and α disagree constitutes a sensitive block for f on $x^{(i)}$. Therefore, there exists a subset $S \subset T$ which is a minimal sensitive block of f on $x^{(i)}$. We then choose $x^{(i+1)} = (x^{(i)})^S$.

Note that the number of bits where $x^{(i+1)}$ agrees with α is strictly greater than the number of bits where $x^{(i)}$ agrees with α .

First we note that since each step consists of flipping a minimal sensitive block, each of the subfunctions $f_{\beta^{(i)}}$ is either AND or NAND: Recall that the subfunction $f_{\beta^{(i)}}$ depends on the bits where $x^{(i)}$ and $x^{(i+1)}$ differ. So for example, if $f(x^{(i)}) = 0$, then the subfunction $f_{\beta^{(i)}}$ is 0 everywhere except when all its free variables agree with $x^{(i+1)}$.

Further, since an AND-NAND path is also an alternating path, the value of $f(x^{(i)})$ alternates between 0 and 1. Therefore, the above described procedure to construct the AND-NAND path alternates between case 1 and case 2.

Also, as noted before, the number of bits where $x^{(i+1)}$ agrees with α is strictly greater than the number of bits where $x^{(i)}$ agrees with α in case 2, whereas in case 1, we can guarantee that this number does not decrease. Since the procedure alternates between the two cases, $x^{(i+2)}$ must agree with α on at least one more bit than $x^{(i)}$, for $i \in \{0, 1, \dots, t-2\}$. Therefore, the procedure must terminate in at most $2|\alpha|$ steps, implying that $t \leq 2|\alpha|$. ◀

Next, note that since each of our measures is upper bounded by certificate complexity (as we proved above), known upper bounds on certificate complexity imply that for each class \mathcal{H} considered in this paper, we have $dia_{\mathcal{H}}(f) \leq O(s(f)^5)$ (using Huang's result [17]) and $dia_{\mathcal{H}}(f) \leq O(deg(f)^3)$ by [24].

Improving these upper bounds would have interesting consequences, as we described in the introduction.

3.2 Upper bounds on certificate complexity in terms of diameters

► **Lemma 3.5.** *For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ and input $x \in \{0, 1\}^n$, we have:*

$$C(f, x) \leq dia_s(f, x) \cdot s(f) \leq dia_{\wedge}(f, x) \cdot s(f) \quad (3)$$

$$C(f, x) \leq dia_{deg}(f, x) \cdot deg(f) \leq dia_{\wedge}(f, x) \cdot deg(f) \quad (4)$$

$$C(f, x) \leq dia_{deg_2}(f, x) \cdot deg_2(f) \leq dia_{\wedge}(f, x) \cdot deg_2(f) \quad (5)$$

Proof. We shall first prove inequality 3.

Let α be a certificate of f on x and $x^{(0)}, x^{(1)}, \dots, x^{(t)}$ be a corresponding full sensitivity path from \bar{x} to α that achieves the minimum value of $dist_{f,s}(\bar{x}, \alpha)$.

31:10 Diameter vs. Certificate Complexity

Then note that every fixed bit of the certificate α must be contained in $\beta^{(i)}$ for some $i \in \{0, 1, \dots, t-1\}$ since $x^{(t)}$ agrees with α and $x^{(0)}$ (i.e. \bar{x}) disagrees with all the fixed bits of α .

Therefore, $|\beta^{(0)}| + |\beta^{(1)}| + \dots + |\beta^{(t-1)}| \geq |\alpha|$. So there must exist an $i \in \{0, 1, \dots, t-1\}$ such that $|\beta^{(i)}| \geq \frac{|\alpha|}{t}$. Now consider the subfunction $f_{\beta^{(i)}}$. Since we considered a full sensitivity path, this subfunction has sensitivity $|\beta^{(i)}|$. Therefore, $s(f) \geq |\beta^{(i)}| \geq \frac{|\alpha|}{t}$.

This implies that $s(f) \cdot \text{dist}_{f,s}(\bar{x}, \alpha) \geq |\alpha|$.

Taking the minimum over all the certificates for f on x gives the first part of inequality 3. The second part follows due to inequality 1. The other two inequalities follow by an analogous argument. ◀

Lemma 3.5 immediately implies the following two theorems.

► **Theorem 3.6.** *For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we have:*

1. $C(f) \leq \text{dia}_s(f) \cdot s(f) \leq \text{dia}_\wedge(f) \cdot s(f)$
2. $C(f) \leq \text{dia}_{\text{deg}}(f) \cdot \text{deg}(f) \leq \text{dia}_\wedge(f) \cdot \text{deg}(f)$
3. $C(f) \leq \text{dia}_{\text{deg}_2}(f) \cdot \text{deg}_2(f) \leq \text{dia}_\wedge(f) \cdot \text{deg}_2(f)$.

Proof. Let x be the input achieving $C(f, x) = C(f)$. Then, equation 3 gives:

$$\begin{aligned} C(f) &= C(f, x) \\ &\leq \text{dia}_s(f, x) \cdot s(f) \\ &\leq \text{dia}_s(f) \cdot s(f). \end{aligned}$$

This gives the first part of the first statement of the theorem, the second part follows by equation 1. The other statements follow similarly from Lemma 3.5 and equation 2. ◀

► **Theorem 3.7.** *For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we have:*

1. $C_{\min}(f) \leq \text{dia}_{\min,s}(f) \cdot s(f) \leq \text{dia}_{\min,\wedge}(f) \cdot s(f)$
2. $C_{\min}(f) \leq \text{dia}_{\min,\text{deg}}(f) \cdot \text{deg}(f) \leq \text{dia}_{\min,\wedge}(f) \cdot \text{deg}(f)$
3. $C_{\min}(f) \leq \text{dia}_{\min,\text{deg}_2}(f) \cdot \text{deg}_2(f) \leq \text{dia}_{\min,\wedge}(f) \cdot \text{deg}_2(f)$.

Proof. Let x be the input for which the minimum value of $\text{dia}_s(f, x)$ is achieved. Then, equation 3 gives:

$$\begin{aligned} C_{\min}(f) &\leq C(f, x) \\ &\leq \text{dia}_s(f, x) \cdot s(f) \\ &= \text{dia}_{\min,s}(f) \cdot s(f). \end{aligned}$$

The first statement of the theorem follows.

Similarly equations 4 and 5, respectively, imply the second and third statements of the theorem. ◀

4 Results for families of functions with small diameters

4.1 Transitive functions with small diameters

In this section, we improve the lower bounds on sensitivity of transitive functions with constant alternating number that follow from the work of Lin and Zhang [20] and then also prove a similar result for transitive functions with constant AND diameter. We then proceed to prove lower bounds on block sensitivity of transitive functions with constant minimum AND diameter.

► **Lemma 4.1.** *For any non-constant transitive function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ the following two statements hold:*

$$s(f)^2 \cdot \text{alt}(f) \geq n$$

$$s(f) \cdot \text{deg}_2(f) \cdot \text{alt}(f) \geq n$$

Proof. Recall that Lemma 2.11 gives that:

$$C(f, 0^n) s(f) \geq n$$

The first part of Lemma 2.13 gives that:

$$C(f, 0^n) \leq \text{alt}(f) \cdot s(f)$$

Together they imply the first statement of the lemma.

The second statement follows similarly using the second part of Lemma 2.13. ◀

Note that this implies $s(f) \geq \sqrt{n}$ for transitive functions with constant alternating number, giving the best possible bound for such functions.

The first statement of Lemma 4.1 is tight for the TRIBES function on n variables: TRIBES is monotone and therefore has $\text{alt}(f) = 1$. Also, TRIBES is transitive as noted in [30]. It is easy to see that TRIBES has $s(f) = \sqrt{n}$.

► **Lemma 4.2.** *For any non-constant transitive function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ the following two statements hold:*

$$s(f)^2 \cdot \text{dia}_\wedge(f) \geq n,$$

$$s(f) \cdot \text{deg}_2(f) \cdot \text{dia}_\wedge(f) \geq n.$$

Proof. Recall that Lemma 2.11 gives that:

$$C(f, 0^n) s(f) \geq n$$

Further equation 3 gives that:

$$C(f, 0^n) \leq \text{dia}_\wedge(f, 0^n) \cdot s(f)$$

Therefore, we get:

$$\text{dia}_\wedge(f, 0^n) \cdot s(f)^2 \geq n$$

which implies the first part of the lemma.

A similar argument gives the second part of the lemma using equation 5. ◀

► **Corollary 4.3.** *For any non-constant transitive function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ with constant alternating number or constant AND diameter $\text{dia}_\wedge(f) = O(1)$, we have:*

$$s(f) \geq \Omega(\sqrt{n})$$

We now prove a lower bound on the block sensitivity of transitive functions under the weaker condition of having constant minimum AND diameter.

► **Lemma 4.4.** *For any non-constant transitive function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ with $\text{dia}_{\min, \wedge}(f) = O(1)$, we have:*

$$bs(f) \geq \Omega(n^{3/7})$$

Proof. We have two cases:

31:12 Diameter vs. Certificate Complexity

Case 1: $C_{min}(f) \geq n^{3/7}$. The first part of Theorem 3.7 implies that $C_{min}(f) \leq dia_{min,\wedge}(f) \cdot s(f) \leq O(s(f))$, since $dia_{min,\wedge}(f) = O(1)$.

Then $n^{3/7} \leq C_{min}(f) \leq O(s(f)) \leq O(bs(f))$ and we are done.

Case 2: $C_{min}(f) \leq n^{3/7}$. Now we use Lemma 2.12 with $r = 7$. This implies that if any transitive function f has $C_{min}(f) \leq n^{3/7}$, then $bs(f) \geq n^{3/7}$, implying the statement of the Lemma in this case. \blacktriangleleft

4.2 Implications to the log-rank conjecture for XOR functions

For any $f: \{0,1\}^n \rightarrow \{0,1\}$, the corresponding XOR function $f \circ \oplus: \{0,1\}^{2n} \rightarrow \{0,1\}$ is defined as: $f \circ \oplus(x, y) = f(x \oplus y)$, where $x \oplus y$ is the bitwise XOR of $x, y \in \{0,1\}^n$.

Lin and Zhang [20] proved that the log-rank conjecture holds for XOR functions $f \circ \oplus$ such that $alt(f)$ is at most polynomial in $\log \|\hat{f}\|_0$ (see Theorem A.11).

In this section, we prove that the log-rank conjecture holds for functions of the form $f(x \oplus y)$ such that the \mathbb{F}_2 -degree diameter of f is upper bounded by a polynomial in the logarithm of the Fourier sparsity of f .

First we prove an analogous result to Theorem A.11 implying the log-rank conjecture for XOR functions with bounded \mathbb{F}_2 -degree diameter. In contrast to the proof of Theorem A.11, we do not need to upper bound $C_{min}^{clo}(f)$, since Theorem 3.6 proves an upper bound directly on $C(f)$ in terms of the product of $dia_{deg_2}(f)$ and $deg_2(f)$ for any function f . This gives us the following result confirming the log-rank conjecture for functions $f \circ \oplus$ with $dia_{deg_2}(f)$ upper bounded by a polynomial in the logarithm of the Fourier sparsity of f :

► **Theorem 4.5.** *For any function $f: \{0,1\}^n \rightarrow \{0,1\}$, we have:*

$$CC(f \circ \oplus) \leq 2dia_{deg_2}(f) \log^2 rank(M_{f \circ \oplus})$$

Proof. Recall the third statement of Theorem 3.6 which states that:

$$C(f) \leq dia_{deg_2}(f) \cdot deg_2(f)$$

Along with Lemma A.10, we get that:

$$\begin{aligned} CC(f \circ \oplus) &\leq 2C(f) \cdot \log rank(M_{f \circ \oplus}) \\ &\leq 2dia_{deg_2}(f) \cdot deg_2(f) \cdot \log rank(M_{f \circ \oplus}) \\ &\leq 2dia_{deg_2}(f) \cdot \log^2 rank(M_{f \circ \oplus}) \end{aligned}$$

Here the last inequality follows from Lemmas A.6 and A.9. \blacktriangleleft

We note that the statement of Theorem 4.5 also holds with $dia_{\wedge}(f)$ replacing $dia_{deg_2}(f)$, since the \mathbb{F}_2 -degree diameter is upper bounded by the AND diameter for any function f as noted in equation 2. We state Theorem 4.5 with $dia_{deg_2}(f)$ instead of $dia_{\wedge}(f)$ because it gives a stronger statement since there exist functions with dia_{deg_2} much smaller than dia_{\wedge} as illustrated by Example 5.2.

Next we prove a common strengthening of Theorem A.11 and Theorem 4.5. We show that the communication complexity of a function of the form $f(x \oplus y)$ can also be upper bounded in terms of the closure of its min \mathbb{F}_2 -degree diameter and the square of the log of rank of its communication matrix $M_{f \circ \oplus}$.

► **Theorem 4.6.** *For any function $f: \{0,1\}^n \rightarrow \{0,1\}$:*

$$CC(f \circ \oplus) \leq 2dia_{min,deg_2}^{clo}(f) \cdot \log^2 rank(M_{f \circ \oplus})$$

Proof. From the third statement of Theorem 3.7, we have that:

$$C_{min}^{clo}(f) \leq dia_{min,deg_2}^{clo}(f) \cdot deg_2(f)$$

Combining this with Lemmas A.6, A.9 and A.10 gives the result. ◀

As we shall note in Lemma 4.7, $dia_{min,deg_2}(f) \leq dia_{min,\wedge}(f) \leq alt(f)$. Since alternating number is a downward non-increasing measure, this implies that $dia_{min,deg_2}^{clo}(f) \leq alt(f)$. Furthermore, Example 5.4 gives a family of functions $f: \{0,1\}^n \rightarrow \{0,1\}$ with alternating number exponentially larger than all our diameters. This shows that Theorem 4.6 is a strictly stronger statement than Theorem A.11.

By definition, we have that for any function f , $dia_{min,deg_2}^{clo}(f) \leq dia_{deg_2}(f)$. Therefore, Theorem 4.6 is a potentially stronger statement than Theorem 4.5. As of now, we are not aware of any example function f separating $dia_{min,deg_2}^{clo}(f)$ from $dia_{deg_2}(f)$. However, we remark that the TRIBES function separates $dia_{min,\wedge}^{clo}(f)$ from $dia_{\wedge}(f)$ as noted in Section 5.8.

We illustrate with examples in Section 5 that, in general, the alternating number of a function and its \mathcal{H} -diameter are incomparable for the different classes \mathcal{H} that we consider. However, in the following lemma, we show that the min AND diameter, and consequently, the min \mathcal{H} -diameter for all our different classes \mathcal{H} , are upper bounded by the alternating number.

► **Lemma 4.7.** *For any function $f: \{0,1\}^n \rightarrow \{0,1\}$, we have,*

$$dia_{min,\wedge}(f) \leq alt(f)$$

Proof. We prove the following relationship, which immediately implies the statement of the lemma:

$$dia_{\wedge}(f, 1^n) \leq alt(f) \tag{6}$$

Let $alt(f) = t$ and let $\mathcal{Q} = x^{(0)}, x^{(1)}, \dots, x^{(t)}$ be an alternating path of length t .

Now, we construct an AND-NAND path $\mathcal{Q}' = z^{(0)}, z^{(1)}, \dots, z^{(t)}$ from 0^n to a certificate α of 1^n in the following way:

Let $z^{(0)} = x^{(0)} = 0^n$. Let $z^{(1)} \leq x^{(1)}$ be a minimal element such that $f(z^{(1)}) = f(x^{(1)})$. Recall that by the definition of alternating path, $f(x^{(i)}) \neq f(x^{(i+1)})$. Thus, the set of variables where $z^{(0)}$ and $z^{(1)}$ differ forms a minimal sensitive block for f on $z^{(0)}$: for all $y \neq z^{(1)}$ such that $z^{(0)} \prec y \prec z^{(1)}$, $f(y) = f(z^{(0)})$ but $f(z^{(1)}) \neq f(z^{(0)})$.

In general, for $i \in \{0, 1, \dots, t-1\}$, let $z^{(i+1)} \prec x^{(i+1)}$ be a minimal element such that $f(z^{(i+1)}) = f(x^{(i+1)})$, and $z^{(i)} \prec z^{(i+1)}$. Thus, the set of variables where $z^{(i)}$ and $z^{(i+1)}$ differ forms a minimal sensitive block for f on $z^{(i)}$. As we noted before (see Section 3.1) subfunctions over a set of variables that forms a minimal sensitive block are either the AND or the NAND function. Thus, for the path \mathcal{Q}' , each subfunction f_{β^i} is either an AND or a NAND.

We will thus get an AND-NAND path $z^{(0)}, z^{(1)}, \dots, z^{(t)}$ of length t . Note that $z^{(t)}$ must agree with some certificate of 1^n , since otherwise, we can get an alternating path for f of length greater than t in the following way: consider the alternating path $\mathcal{Q}' = z^{(0)}, z^{(1)}, \dots, z^{(t)}$. Since $z^{(t)}$ does not agree with any certificate of 1^n , the partial assignment α defined as $\alpha_i = 1$ whenever $z_i^{(t)} = 1$ and $\alpha_i = \star$ otherwise, is not a certificate of f . This implies the existence of an input $z^{(t+1)}$ such that $z^{(t)} \prec z^{(t+1)}$ and $f(z^{(t)}) \neq f(z^{(t+1)})$. Therefore, the path $\mathcal{Q}'' = z^{(0)}, z^{(1)}, \dots, z^{(t)}, z^{(t+1)}$ is an alternating path of length $t+1$ contradicting the fact that $alt(f) = t$.

Therefore, the path $z^{(0)}, z^{(1)}, \dots, z^{(t)}$ is an AND-NAND path from 0^n to some certificate of f on 1^n and the statement follows. ◀

5 Separating Examples

In this section, we give several examples, separating various types of diameters from alternating number and from each other.

5.1 Separating $dia_s(f)$ from $dia_{\wedge}(f)$

The following example has constant sensitivity diameter, whereas its AND diameter equals the number of input variables.

► **Example 5.1.** Let $f: \{0,1\}^n \rightarrow \{0,1\}$ be the PARITY function on n bits i.e. $PARITY(x) = \bigoplus_{i \in [n]} x_i$.

It is easy to see that $dia_{\wedge}(PARITY) = n$, since for any partial assignment α , the subfunction $PARITY_{\alpha}$ belongs to the AND-NAND class only if α fixes all but 1 variable.

On the other hand, for any input $x \in \{0,1\}^n$, $dia_s(PARITY, x) = 1$. This is because we can consider α to be the certificate fixing all the bits of x and then the path \bar{x}, x is a valid \mathcal{H} -path since $s(PARITY) = n$ (i.e. $PARITY$ induced on the “entire cube” has full sensitivity).

5.2 Separating $dia_{deg}(f)$ (and also $dia_{deg_2}(f)$) from $dia_{\wedge}(f)$

The next example has constant values for both its real degree diameter as well as \mathbb{F}_2 -degree diameter, but has large AND diameter.

► **Example 5.2.** Define $f: \{0,1\}^n \rightarrow \{0,1\}$ as follows:

$$\begin{aligned} f(0^n) &= 1, \\ f(x) &= \bigoplus_{i \in [n]} x_i \text{ otherwise.} \end{aligned}$$

It is easy to show that $dia_{\wedge}(f) = n$ by a similar argument as in example 5.1.

Also, we note that $deg_2(f) = n$. This follows from a result of Beigel and Bernasconi [4], stating that for any Boolean function, $deg_2(f) = n$ iff $|f^{-1}(1)|$ is odd.

Therefore, by an analogous argument as in example 5.1, for any input $x \in \{0,1\}^n$, $dia_{deg_2}(f, x) = 1$. (We can again consider α to be the certificate fixing all the bits of x and the path \bar{x}, x is a valid \mathcal{H} -path since $deg_2(f) = n$.)

Therefore, we have $dia_{deg_2}(f) = 1$.

We also have that for any Boolean function f , $deg_2(f) \leq deg(f)$ (see for example, proposition 6.23 in [30]).

Therefore, $deg(f) = n$, and by a similar argument as for the \mathbb{F}_2 -degree, $dia_{deg}(f) = 1$.

We note that the PARITY function from example 5.1 also separates $dia_{deg}(f)$ from $dia_{\wedge}(f)$. This is because $dia_{deg}(PARITY) = 1$ as we discuss below, whereas $dia_{\wedge}(PARITY) = n$ as noted in Example 5.1.

5.3 Separating $dia_{deg}(f)$ from $dia_{deg_2}(f)$

Recall from equation 2 that $dia_{deg}(f) \leq dia_{deg_2}(f)$ for any boolean function f . To separate $dia_{deg}(f)$ from $dia_{deg_2}(f)$, we consider again the PARITY function discussed in Example 5.1. It is easy to see that $deg_2(PARITY) = 1$. Moreover, for any partial assignment $\alpha: [n] \rightarrow \{0, 1, \star\}$, the subfunction $PARITY_{\alpha}$ is also a PARITY function on the free bits and therefore, $deg_2(PARITY_{\alpha}) = 1$. So we have $dia_{deg_2}(PARITY) = n$. However, $deg(PARITY) = n$, by the characterization due to Shi and Yao (see in the survey [9]) which states that for

any function $f: \{0,1\}^n \rightarrow \{0,1\}$, $\text{deg}(f) = n$ iff the number of 1-inputs with an even number of 1's does not equal the number of 1-inputs with an odd number of 1's. Therefore, $\text{dia}_{\text{deg}}(\text{PARITY}) = 1$.

5.4 Separating $\text{dia}_s(f)$ and $\text{dia}_{\text{deg}_2}(f)$ from each other

The measures $\text{dia}_s(f)$ and $\text{dia}_{\text{deg}_2}(f)$ are incomparable, and we provide examples separating them in both directions.

We again revisit the PARITY function considered in Example 5.1 to illustrate a function with small $\text{dia}_s(f)$ and large $\text{dia}_{\text{deg}_2}(f)$. As noted before, $\text{dia}_{\text{deg}_2}(\text{PARITY}) = n$, whereas $\text{dia}_s(\text{PARITY}) = 1$, achieving the required separation.

To illustrate a separation in the other direction, we consider the TRIBES function on n^2 bits that has $\text{dia}_s(\text{TRIBES}) = \Theta(n)$ as we shall see in Example 5.3. However, $\text{deg}_2(\text{TRIBES}) = n^2$ and therefore, $\text{dia}_{\text{deg}_2}(\text{TRIBES}) = 1$.

5.5 Separating $\text{dia}_s(f)$ from $\text{dia}_{\text{deg}}(f)$

We now mention a separating example with $\text{dia}_{\text{deg}}(f)$ much smaller than $\text{dia}_s(f)$. The TRIBES function (see also Example 5.3) on n^2 bits has $\text{dia}_s(\text{TRIBES}) = \Theta(n)$. However, $\text{deg}(\text{TRIBES}) = n^2$ and therefore, $\text{dia}_{\text{deg}}(\text{TRIBES}) = 1$.

We are not aware of any examples as yet, separating these measures in the other direction. We do believe these measures to be incomparable, and it would be an interesting exercise to find functions f with $\text{dia}_s(f)$ asymptotically smaller than $\text{dia}_{\text{deg}}(f)$.

5.6 Example with $\text{alt}(f)$ smaller than $\text{dia}_s(f)$ (and also $\text{dia}_\wedge(f)$)

We now present an example where the alternating number is much smaller than $\text{dia}_\wedge(f)$ as well as $\text{dia}_s(f)$.

► **Example 5.3.** Consider the TRIBES function $f: \{0,1\}^{n^2} \rightarrow \{0,1\}$ defined as:

$$\text{TRIBES}(x_{11}, x_{12}, \dots, x_{1n}, x_{21}, \dots, x_{2n}, \dots, x_{n1}, \dots, x_{nn}) = \bigvee_{i \in [n]} \bigwedge_{j \in [n]} x_{ij}$$

We first note that since TRIBES is monotone, $\text{alt}(\text{TRIBES}) = 1$.

We shall show that $\text{dia}_\wedge(\text{TRIBES}) \geq 2n - 1$.

Consider the 1-input $x = 1^n(0^2 1^{n-2})^{n-1}$. In other words, the first block of n bits of x are set to 1, the remaining $n - 1$ blocks of n bits each have the first 2 bits set to 0 and the rest set to 1.

Note that $\text{TRIBES}(x) = 0$.

Now, let $x^{(0)}, x^{(1)}, \dots, x^{(t)}$ be any valid AND-NAND path from x to α where α is a certificate of \bar{x} . (Note that we have switched the roles of x and \bar{x} in this example for convenience.)

Since $\text{TRIBES}(x^{(0)}) \neq f(x^{(1)}) = 0$, the partial assignment $\beta^{(0)}$ must have free bits belonging to the first block of n bits, and moreover, it cannot contain any free bits from any of the other blocks, in order for the function $\text{TRIBES}_{\beta^{(0)}}$ to belong to the AND-NAND class.

For the next step, since $\text{TRIBES}(x^{(2)}) = 1$, it is necessary that both 0-bits must be flipped from one of the remaining blocks. In other words, the partial assignment $\beta^{(1)}$ must have exactly two free variables corresponding to the first two bits of some block (other than the first block).

Again, as before, the partial assignment $\beta^{(2)}$ must contain free variables from the block which now only contains 1-bits.

In this way, any shortest valid AND-NAND path must alternate between changing some block to contain only 1-bits (thereby changing the function value to 1), and then flipping some 1-bit in that block to 0 (changing the function value to 0).

Eventually, every block will contain a bit that was flipped from a 1 to a 0, and the set of these bits shall constitute a certificate of \bar{x} .

Therefore, we have that $\text{dia}_{\wedge}(\text{TRIBES}) \geq 2n - 1$. We note that this bound is tight up to constant factors, as can be seen from Theorem B.1 which implies that $\text{dia}_{\wedge}(\text{TRIBES}) \leq O(n)$.

A similar argument also works to show that $\text{dia}_s(\text{TRIBES}) \geq 2n - 1$, and therefore, $\text{dia}_s(\text{TRIBES}) = \theta(n)$ due to Theorem B.1.

However, since $\text{deg}(\text{TRIBES}) = n^2$, we have that $\text{dia}_{\text{deg}}(\text{TRIBES}) = 1$.

Similarly, $\text{deg}_2(\text{TRIBES}) = n^2$ and therefore, $\text{dia}_{\text{deg}_2}(\text{TRIBES}) = 1$.

5.7 Example with $\text{alt}(f)$ larger than all our diameters

We refer to an example from [13] that separates $\text{alt}(f)$ from $C(f)$ (and consequently, from all of our diameters).

► **Example 5.4.** Let f be the function constructed in [13] as an example where $\text{alt}(f)$ is exponentially larger than $DT(f)$ and therefore also $C(f)$ (since $DT(f) \geq C(f)$). We note that since $\text{dia}_{\wedge}(f) \leq 2C(f)$ due to Lemma 3.4, f also acts as a separating example where $\text{alt}(f)$ is exponentially larger than $\text{dia}_{\wedge}(f)$, and therefore, also exponentially larger than $\text{dia}_s(f)$, $\text{dia}_{\text{deg}_2}(f)$ and $\text{dia}_{\text{deg}}(f)$.

5.8 Separating $\text{dia}_{\min, \wedge}^{\text{clo}}(f)$ from $\text{dia}_{\wedge}(f)$ and $\text{dia}_{\min, s}^{\text{clo}}(f)$ from $\text{dia}_s(f)$

For separating $\text{dia}_{\min, \wedge}^{\text{clo}}(f)$ from $\text{dia}_{\wedge}(f)$, we revisit the TRIBES function considered in Example 5.3. As noted in that example, $\text{dia}_{\wedge}(\text{TRIBES}) = \Theta(n)$. On the other hand, we have that $\text{dia}_{\min, \wedge}^{\text{clo}}(\text{TRIBES}) \leq \text{alt}(\text{TRIBES}) = 1$, thereby achieving the required separation. The same separation is also achieved for the TRIBES function between $\text{dia}_{\min, s}^{\text{clo}}(f)$ and $\text{dia}_s(f)$ by an analogous argument.

References

- 1 Scott Aaronson, Shalev Ben-David, Robin Kothari, and Avishay Tal. Quantum Implications of Huang’s Sensitivity Theorem. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:66, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/066>.
- 2 Kazuyuki Amano. Minterm-transitive functions with asymptotically smallest Block Sensitivity. *Inf. Process. Lett.*, 111(23-24):1081–1084, 2011. doi:10.1016/j.ipl.2011.09.008.
- 3 Kaspars Balodis. Several Separations Based on a Partial Boolean Function. *arXiv e-prints*, March 2021. arXiv:2103.05593.
- 4 Richard Beigel and Anna Bernasconi. A note on the polynomial representation of boolean functions over GF(2). *International Journal of Foundations of Computer Science*, 10(04):535–542, 1999. doi:10.1142/S012905419900037X.
- 5 Shalev Ben-David, Mika Goos, Siddhartha Jain, and Robin Kothari. Unambiguous DNFs from HEX. *arXiv e-prints*, February 2021. arXiv:2102.08348.

- 6 Shalev Ben-David, Pooya Hatami, and Avishay Tal. Low-Sensitivity Functions from Unambiguous Certificates. In *Proceedings of Innovations in Theoretical Computer Science Conference (ITCS)*, pages 28:1–28:23, 2017. doi:10.4230/LIPIcs.ITCS.2017.28.
- 7 A. Bernasconi and B. Codenotti. Spectral analysis of boolean functions as a graph eigenvalue problem. *IEEE Transactions on Computers*, 48(3):345–351, 1999.
- 8 Eric Blais, Clément L Canonne, Igor C Oliveira, Rocco A Servedio, and Li-Yang Tan. Learning circuits with few negations. *arXiv preprint*, 2014. arXiv:1410.8420.
- 9 Harry Buhman and Ronald De Wolf. Complexity Measures and Decision Tree Complexity: A Survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.
- 10 Sourav Chakraborty. On the Sensitivity of Cyclically-Invariant Boolean functions. *Discrete Mathematics & Theoretical Computer Science*, 13(4):51–60, 2011. URL: <http://dmtcs.episciences.org/552>.
- 11 J.-C Chang and H.-L Wu. The log-rank conjecture for read-k xor functions. *Journal of Information Science and Engineering*, 34:391–399, March 2018.
- 12 Siddhesh Chabhal and Anna Gál. Tight bounds on sensitivity and block sensitivity of some classes of transitive functions. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:134, 2020. URL: <https://ecc.ecc.weizmann.ac.il/report/2020/134/>.
- 13 Krishnamoorthy Dinesh and Jayalal Sarma. Alternation, Sparsity and Sensitivity: Combinatorial Bounds and Exponential Gaps. In *Proceedings of the 4th International Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, pages 260–273, 2018. doi:10.1007/978-3-319-74180-2_22.
- 14 Krishnamoorthy Dinesh and Jayalal Sarma. Sensitivity, affine transforms and quantum communication complexity. In *Computing and Combinatorics - 25th International Conference, COCOON 2019, Proceedings*, volume 11653 of *Lecture Notes in Computer Science*, pages 140–152. Springer, 2019.
- 15 Andrew Drucker. Block Sensitivity of Minterm-Transitive functions. *Theor. Comput. Sci.*, 412(41):5796–5801, 2011. doi:10.1016/j.tcs.2011.06.025.
- 16 Siyao Guo, Tal Malkin, Igor C. Oliveira, and Alon Rosen. The power of negations in cryptography. In *Theory of Cryptography*, pages 36–65, 2015.
- 17 Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019.
- 18 Raghav Kulkarni and Miklos Santha. Query complexity of matroids. In Paul G. Spirakis and Maria Serna, editors, *Algorithms and Complexity*, pages 300–311, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- 19 Sophie Laplante, Reza Naserasr, and Anupa Sunny. Sensitivity Lower Bounds from Linear Dependencies. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:14, 2020. doi:10.4230/LIPIcs.MFCS.2020.62.
- 20 Chengyu Lin and Shengyu Zhang. Sensitivity Conjecture and Log-Rank Conjecture for Functions with Small Alternating Numbers. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 51:1–51:13, 2017. doi:10.4230/LIPIcs.ICALP.2017.51.
- 21 László Lovász and Michael Saks. Lattices, mobius functions and communications complexity. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 81–90. IEEE Computer Society, 1988.
- 22 A. A. Markov. On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, October 1958. doi:10.1145/320941.320945.
- 23 Kurt Mehlhorn and Erik M. Schmidt. Las vegas is better than determinism in vlsi and distributed computing (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC '82*, page 330–337. Association for Computing Machinery, 1982. doi:10.1145/800070.802208.

- 24 Gatis Midrijanis. Exact quantum query complexity for total boolean functions. *arXiv preprint*, 2004. [arXiv:quant-ph/0403168](https://arxiv.org/abs/quant-ph/0403168).
- 25 Ashley Montanaro and Tobias Osborne. On the communication complexity of XOR functions. *CoRR*, abs/0909.3392, 2009. [arXiv:0909.3392](https://arxiv.org/abs/0909.3392).
- 26 Hiroki Morizumi. Limiting negations in formulas. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, page 701–712. Springer-Verlag, 2009. doi:10.1007/978-3-642-02927-1_58.
- 27 Hiroki Morizumi. Limiting negations in non-deterministic circuits. *Theoretical Computer Science*, 410(38):3988–3994, 2009. doi:10.1016/j.tcs.2009.05.018.
- 28 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, December 1994. doi:10.1007/BF01263419.
- 29 Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Comb.*, 15(4):557–565, 1995. doi:10.1007/BF01192527.
- 30 Ryan O’Donnell. *Analysis of Boolean functions*. Cambridge University Press, 2014.
- 31 Miklos Santha and Christopher Wilson. Limiting negations in constant depth circuits. *SIAM Journal on Computing*, 22(2):294–302, 1993. doi:10.1137/0222022.
- 32 Xiaoming Sun. Block Sensitivity of Weakly Symmetric Functions. In *Proceedings of the Third International Conference on Theory and Applications of Models of Computation (TAMC)*, pages 339–344, 2006. doi:10.1007/11750321_32.
- 33 Xiaoming Sun. An improved lower bound on the Sensitivity Complexity of Graph Properties. *Theor. Comput. Sci.*, 412(29):3524–3529, 2011. doi:10.1016/j.tcs.2011.02.042.
- 34 Shao Chin Sung and Keisuke Tanaka. Limiting negations in bounded-depth circuits: An extension of markov’s theorem. In *Algorithms and Computation*, pages 108–116. Springer, 2003.
- 35 H. Y. Tsang, C. H. Wong, N. Xie, and S. Zhang. Fourier sparsity, spectral norm, and the log-rank conjecture. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 658–667, 2013.
- 36 Hing Yin Tsang. On boolean functions with low sensitivity. *Manuscript*, 2014. URL: <http://theorycenter.cs.uchicago.edu/REU/2014/final-papers/tsang.pdf>.
- 37 Hing Yin Tsang, Chung Hoi Wong, Ning Xie, and Shengyu Zhang. Fourier Sparsity, Spectral Norm, and the Log-Rank Conjecture. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 658–667, 2013. doi:10.1109/FOCS.2013.76.
- 38 Gyorgy Turan. The Critical Complexity of Graph Properties. *Information Processing Letters*, 18(3):151–153, 1984. doi:10.1016/0020-0190(84)90019-X.
- 39 Zhiqiang Zhang and Yaoyun Shi. Communication complexities of symmetric xor functions. *Quantum Info. Comput.*, 9(3):255–263, 2009.

A Further Background

► **Definition A.1** (Real degree of a function). *A polynomial p over the reals with n variables is said to represent a Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ if $p(x) = f(x)$ for all $x \in \{0, 1\}^n$. The degree of the unique multilinear polynomial over the reals representing f is defined to be the degree of f , and is denoted as $\deg(f)$.*

► **Definition A.2** (\mathbb{F}_2 degree of a function). *For any Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, the degree of the unique multilinear polynomial over \mathbb{F}_2 representing f is called as the \mathbb{F}_2 degree of f , denoted $\deg_2(f)$.*

► **Definition A.3** (Decision tree complexity). *A decision tree evaluating a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ is a rooted binary tree with internal nodes labeled by variables and leaves labeled by $\{0, 1\}$. The depth of a decision tree is defined as the length of a longest root to leaf path in the tree.*

For any input $x \in \{0, 1\}^n$, the label at the leaf reached by following the decision tree queries is the evaluation of the decision tree on x . The decision tree is said to compute the function f if it evaluates to $f(x)$ on every input $x \in \{0, 1\}^n$. The decision tree complexity of a boolean function f , denoted $DT(f)$, is defined as the smallest possible depth of any decision tree computing f .

► **Definition A.4** (Fourier sparsity of a function). Any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ can be uniquely represented as

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) (-1)^{\sum_{i \in S} x_i}$$

This is said to be the Fourier expansion of f and the coefficients $\hat{f}(S)$ are called the Fourier coefficients of f . The number of non-zero Fourier coefficients of f is defined to be the Fourier sparsity of f , denoted $\|\hat{f}\|_0$.

We note that usually the Fourier representation is considered for functions of the form $f: \{0, 1\}^n \rightarrow \{+1, -1\}$. We use this version of the definition because it exactly captures the rank of the communication matrix for XOR functions (see Section A.1).

A.1 The approach of Lin and Zhang

We review the approach of Lin and Zhang [20] which we build upon in Section 4.2.

We first define the closure of min Certificate Complexity which was implicit in the approach of [35], and defined in [20]:

► **Definition A.5** (Closure of min Certificate Complexity; [35, 20]). For a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we define the closure of the min certificate complexity of f as:

$$C_{min}^{clo}(f) = \max_{\alpha} C_{min}(f_{\alpha})$$

where the maximum is taken over all possible partial assignments α on n variables, or in other words, over all possible subfunctions f_{α} of f .

Note that $C_{min}^{clo}(f) \leq C(f)$.

They also note that it is possible to define the closure for any complexity measure $M(f)$ for function f as $M^{clo}(f) = \max_{\alpha} M(f_{\alpha})$, where the maximum is taken over all subfunctions f_{α} of f . A measure M is said to be downward non-increasing if for any function f , it holds that $M(f_{\alpha}) \leq M(f)$ for any subfunction f_{α} of f . Note that the measures sensitivity, block sensitivity, certificate complexity, decision tree complexity, \mathbb{F}_2 -degree, Fourier sparsity, alternating number are all downward non-increasing. It follows from the definition of downward non-increasing measures that whenever measure M is downward non-increasing, it holds that $M^{clo}(f) = M(f)$.

We first note the following result from [7] stating that the rank of the communication matrix $M_{f \circ \oplus}$ exactly equals the Fourier sparsity of f :

► **Lemma A.6** ([7]). For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we have: $\text{rank}(M_{f \circ \oplus}) = \|\hat{f}\|_0$.

Next, we note the following lemma about communication complexity of XOR functions:

► **Lemma A.7** ([25]). For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we have: $CC(f \circ \oplus) \leq 2DT(f)$.

31:20 Diameter vs. Certificate Complexity

The following lemma upper bounds the decision tree complexity in terms of the product of its certificate complexity and \mathbb{F}_2 -degree. We note that the second inequality follows because $C_{min}^{clo}(f) \leq C(f)$.

► **Lemma A.8** ([36, 37, 20]). *For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we have:*

$$DT(f) \leq C_{min}^{clo}(f) \cdot deg_2(f) \leq C(f) \cdot deg_2(f)$$

The \mathbb{F}_2 -degree of any function is upper bounded by the logarithm of its Fourier sparsity as proved in [7]:

► **Lemma A.9** ([7]). *For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we have: $deg_2(f) \leq \log \|\hat{f}\|_0$.*

Lemmas A.6, A.7, A.8, A.9 imply the following result as also noted in [20]:

► **Lemma A.10** ([20]). *For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$:*

$$CC(f \circ \oplus) \leq 2C_{min}^{clo}(f) \cdot \log rank(M_{f \circ \oplus}) \leq 2C(f) \cdot \log rank(M_{f \circ \oplus})$$

It follows from Lemma A.10 that proving an upper bound on the certificate complexity of a function f in terms of a polynomial in $\log rank(M_{f \circ \oplus})$ would imply the log-rank conjecture for the corresponding XOR function $f \circ \oplus$.

Another approach towards proving the log-rank conjecture for XOR functions would be to upper bound $C_{min}^{clo}(f)$ directly. One way to achieve this for a class of functions would be to prove an upper bound of the form $C_{min}(f) \leq M(f)$ where $M(f)$ is a complexity measure that is downward non-increasing, since that would imply the bound $C_{min}^{clo}(f) \leq M(f)$, thereby proving the log-rank conjecture for functions $f \circ \oplus$ with bounded value of $M(f)$. Lin and Zhang [20] use this approach to prove that the log-rank conjecture holds for XOR functions $f \circ \oplus$ which are such that $alt(f)$ is at most polynomial in $\log \|\hat{f}\|_0$.

In particular, they achieve this by proving that for every boolean function f , $C_{min}(f) \leq alt(f)deg_2(f)$. Since both $alt(f)$ and $deg_2(f)$ are downward non-increasing, they get the following result:

► **Theorem A.11** (Theorem 2 from [20]). *For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, we have:*

$$CC(f \circ \oplus) \leq 2alt(f) \cdot \log^2 rank(M_{f \circ \oplus})$$

B Diameter under OR-composition

In this section, we study the behavior of diameters under OR-composition. In particular, we prove that any diameter of the OR-composition of two functions is upper bounded by a sum of their corresponding individual diameters (plus a constant factor).

In what follows, for two functions $f, g: \{0, 1\}^n \rightarrow \{0, 1\}$, we define the OR-composed function $f \vee g: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ as $f \vee g(x_1, x_2) = 1$ if $f(x_1) = 1$ or $g(x_2) = 1$, and $f \vee g(x_1, x_2) = 0$ otherwise, for $x_1, x_2 \in \{0, 1\}^n$.

We now prove a result relating the diameter of OR-composition of two functions with their individual diameters as mentioned before:

► **Theorem B.1.** *For any functions $f, g: \{0, 1\}^n \rightarrow \{0, 1\}$, we have:*

$$dia_{\wedge}(f \vee g) \leq dia_{\wedge}(f) + dia_{\wedge}(g) + 3$$

Proof. Consider any input $(x_1, x_2) \in \{0, 1\}^{2n}$. We have two cases as follows:

Case 1: $f \vee g(x_1, x_2) = 0$. Consider the AND-NAND path $z_1^{(0)}, z_1^{(1)}, \dots, z_1^{(t_1)}$ (where $\bar{x}_1 = z_1^{(0)}$) achieving $\text{dia}_\wedge(f, x_1)$ and the AND-NAND path $z_2^{(0)}, z_2^{(1)}, \dots, z_2^{(t_2)}$ (where $\bar{x}_2 = z_2^{(0)}$) achieving $\text{dia}_\wedge(g, x_2)$.

We first deal with the case when $f(\bar{x}_1) = 0$ and $g(\bar{x}_2) = 0$. Observe that the following is a valid AND-NAND path for $f \vee g$:

$$(z_1^{(0)}, z_2^{(0)}), (z_1^{(0)}, z_2^{(1)}), \dots, (z_1^{(0)}, z_2^{(t_2)}), (z_1^{(1)}, z_2^{(t_2)}), (z_1^{(2)}, z_2^{(t_2)}), \dots, (z_1^{(t_1)}, z_2^{(t_2)}).$$

This follows because of the simple observation, that for any input $(a, b) \in \{0, 1\}^{2n}$, if $g(b) = 0$, then $f \vee g(a, b) = f(a)$. Due to this observation, in the first t_2 steps of the above path, the input to f is fixed to $z_1^{(0)}$ which is such that $f(z_1^{(0)}) = 0$. Therefore, throughout these steps, we have that $f \vee g(z_1^{(0)}, z_2^{(i)}) = g(z_2^{(i)})$ for any $i \in \{0, 1, \dots, t_2\}$. Since $z_2^{(0)}, z_2^{(1)}, \dots, z_2^{(t_2)}$ is a valid AND-NAND path of g , the first t_2 steps form valid steps of an AND-NAND path of $f \vee g$.

Similarly, since $z_2^{(t_2)}$ agrees with a certificate for g on x_2 , it holds that $g(z_2^{(t_2)}) = 0$. Therefore, for the next t_1 steps of the path, it holds that $f \vee g(z_1^{(i)}, z_2^{(t_2)}) = f(z_1^{(i)})$ for any $i \in \{0, 1, \dots, t_1\}$, and a similar argument goes through as for the first t_2 steps of the path.

Finally, since $z_1^{(t_1)}$ agrees with a certificate for f on x_1 and $z_2^{(t_2)}$ agrees with a certificate for g on x_2 , the input $(z_1^{(t_1)}, z_2^{(t_2)})$ agrees on a certificate for $f \vee g$ on the input (x_1, x_2) . We therefore get a valid AND-NAND path for the function $f \vee g$ of length $t_1 + t_2$.

Now, we consider the case when $f(\bar{x}_1) = 1$ and $g(\bar{x}_2) = 0$. In this case, we first perform an additional step of flipping the bits of any minimal sensitive block B for f on \bar{x}_1 to get to a 0-input of f i.e. \bar{x}_1^B . We then follow the argument of the previous case, and without changing the input to f , take t_2 steps corresponding to the AND-NAND path for g i.e. $z_2^{(0)}, z_2^{(1)}, \dots, z_2^{(t_2)}$. We then “undo” the first step by flipping back the bits of B in the input corresponding to f to get to the input $(\bar{x}_1, z_2^{(t_2)})$. Finally, we take the t_1 steps of the AND-NAND path for f starting from input \bar{x}_1 to get a valid AND-NAND path of total length $t_1 + t_2 + 2$.

Similar argument works for the case when $f(\bar{x}_1) = 0$ and $g(\bar{x}_2) = 1$.

Finally, the case with $f(\bar{x}_1) = 1$ and $g(\bar{x}_2) = 1$ goes through a similar argument, with the difference that the first step involves simultaneously flipping minimal sensitive blocks for both f and g , on the respective inputs \bar{x}_1 and \bar{x}_2 , to get input $(\bar{x}_1^{B_1}, \bar{x}_2^{B_2})$, say. The next step flips back these bits only for g to get the input $(\bar{x}_1^{B_1}, \bar{x}_2)$. The argument then proceeds as in the previous case, where we take the AND-NAND path for g starting from input \bar{x}_2 , followed by flipping back the bits of block B_1 for the input to f , followed by the AND-NAND path for f starting from input \bar{x}_1 .

This gives a valid AND-NAND path of length $t_1 + t_2 + 3$ for $f \vee g$ starting from input (\bar{x}_1, \bar{x}_2) .

Case 2: $f \vee g(x_1, x_2) = 1$. Wlog, assume that $g(x_2) = 1$.

In this case, we follow a similar proof strategy as in Case 1, with the difference that in this case, we only need to follow the steps corresponding to a valid AND-NAND path for g starting from \bar{x}_2 , since in this case, a certificate for g on x_2 would also be a certificate for $f \vee g$ on (x_1, x_2) . Apart from that, we follow a similar argument, where we first flip appropriate blocks of bits, if necessary, to ensure that we are at a 0-input of f . We then follow the AND-NAND path for g to get a certificate for $f \vee g$ on (x_1, x_2) . This gives an AND-NAND path for $f \vee g$ of length at most $t_2 + 2$. ◀

31:22 Diameter vs. Certificate Complexity

We remark that Theorem B.1 also holds for the three other diameters we consider i.e. for $dia_{deg}(f)$, $dia_{deg_2}(f)$ and $dia_s(f)$ by an analogous argument.

Note that the TRIBES function $f: \{0, 1\}^{n^2} \rightarrow \{0, 1\}$ as discussed in Example 5.3 is an OR-composition of n copies of the AND_n function (i.e. the AND function on n bits). Since $dia_{\wedge}(AND_n) = 1$, Theorem B.1 implies the bound: $dia_{\wedge}(TRIBES) \leq \Theta(n)$. As seen in Example 5.3, $dia_{\wedge}(TRIBES) \geq 2n - 1$ and therefore, this bound is asymptotically tight for the TRIBES function.

However, Theorem B.1 does not always give a tight bound for OR-composed functions, as can be seen from the example of the function $OR_n: \{0, 1\}^n \rightarrow \{0, 1\}$ i.e. the OR function on n bits. OR_n is also an OR-composition of n copies of the function $g: \{0, 1\} \rightarrow \{0, 1\}$ with single-bit inputs, where $g(x) = x$. Since $dia_{\wedge}(g) = 1$, Theorem B.1 gives the bound $dia_{\wedge}(OR_n) \leq \Theta(n)$. This bound is not tight since it holds that $dia_{\wedge}(OR_n) = 1$.

Budgeted Dominating Sets in Uncertain Graphs

Keerti Choudhary ✉

Indian Institute of Technology Delhi, India

Avi Cohen ✉

Tel Aviv University, Israel

N. S. Narayanaswamy ✉ 

Department of Computer Science and Engineering, IIT Madras, India

David Peleg ✉ 

Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel

R. Vijayaragunathan ✉ 

Department of Computer Science and Engineering, IIT Madras, India

Abstract

We study the *Budgeted Dominating Set* (BDS) problem on uncertain graphs, namely, graphs with a probability distribution p associated with the edges, such that an edge e exists in the graph with probability $p(e)$. The input to the problem consists of a vertex-weighted uncertain graph $\mathcal{G} = (V, E, p, \omega)$ and an integer *budget* (or *solution size*) k , and the objective is to compute a vertex set S of size k that maximizes the expected total domination (or total weight) of vertices in the closed neighborhood of S . We refer to the problem as the *Probabilistic Budgeted Dominating Set* (PBDS) problem. In this article, we present the following results on the complexity of the PBDS problem.

1. We show that the PBDS problem is NP-complete even when restricted to uncertain *trees* of diameter at most four. This is in sharp contrast with the well-known fact that the BDS problem is solvable in polynomial time in trees. We further show that PBDS is W[1]-hard for the budget parameter k , and under the *Exponential time hypothesis* it cannot be solved in $n^{o(k)}$ time.
2. We show that if one is willing to settle for $(1 - \epsilon)$ approximation, then there exists a PTAS for PBDS on trees. Moreover, for the scenario of uniform edge-probabilities, the problem can be solved optimally in polynomial time.
3. We consider the parameterized complexity of the PBDS problem, and show that Uni-PBDS (where all edge probabilities are identical) is W[1]-hard for the parameter pathwidth. On the other hand, we show that it is FPT in the combined parameters of the budget k and the treewidth.
4. Finally, we extend some of our parameterized results to planar and apex-minor-free graphs.

Our first hardness proof (Thm. 1) makes use of the new problem of k -SUBSET Σ -II MAXIMIZATION (k -SPM), which we believe is of independent interest. We prove its NP-hardness by a reduction from the well-known k -SUM problem, presenting a close relationship between the two problems.

2012 ACM Subject Classification Mathematics of computing \rightarrow Graph algorithms

Keywords and phrases Uncertain graphs, Dominating set, NP-hard, PTAS, treewidth, planar graph

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.32

Related Version *Full Version*: <https://arxiv.org/abs/2107.03020>

Acknowledgements We thank an anonymous reviewer for pointing us to [6], yielding a shorter proof of the FPT algorithm for Uni-PBDS parameterized by treewidth and k .

David Peleg is supported by the Venky Harinarayanan and Anand Rajaraman Visiting Chair Professorship at the Indian Institute of Technology Madras, Chennai, India (IIT Madras). Supported by the chair's funds, this work was done in part when David Peleg, Avi Cohen, and Keerti Choudhary visited IIT Madras and when R. Vijayaragunathan visited the Weizmann Institute of Science, Rehovot, Israel.



© Keerti Choudhary, Avi Cohen, N. S. Narayanaswamy, David Peleg, and R. Vijayaragunathan; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 32; pp. 32:1–32:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Background and Motivation. Many optimization problems in network theory deal with placing resources in key vertices in the network so as to maximize coverage. Some practical contexts where such coverage problems occur include placing mobile towers in wireless networks to maximize reception, assigning emergency vehicle centers in a populated area to guarantee fast response, opening production plants to ensure short distribution lines, and so on. In the context of social networks, the problem of spreading influencers so as to affect as many of the network members as possible has recently attracted considerable interest.

Coverage problems may assume different forms depending on the optimized parameter. A basic “full coverage” variant is the classical *dominating set* problem, which asks to find a *minimum* vertex set S such that each vertex not in S is *dominated by* S , i.e., is adjacent to at least one vertex in S . In the dual *budgeted dominating set (BDS)* problem, given a bound k (the budget), it is required to find a set S of size at most k *maximizing* the number of covered vertices. Over *vertex weighted* graphs, the goal is to maximize the total *weight* of the covered vertices, also known as the *domination*. It is this variant that we’re concerned with here.

Traditionally, coverage problems involve a fixed network of static topology. The picture becomes more interesting when the network structure is uncertain, due to potential edge connections and disconnections or link failures. Pre-selection of resource locations at the design stage becomes more challenging in such partial-information settings.

In this work, we study the problem in one of the most fundamental settings, where the input is a graph whose edges fail independently with a given probability. The goal is to find a k -element set that maximizes the *expected* (1-hop) coverage (or domination). Our results reveal that the probabilistic versions of the coverage problem are significantly harder than their deterministic counterparts, and analyzing them require more elaborate techniques.

An *uncertain graph* \mathcal{G} is a triple (V, E, p) , where V is a set of n vertices, $E \subseteq V \times V$ is a set of m edges, and the function $p : E \rightarrow [0, 1]$ assigns a probability of existence to each edge in E . So an m edge uncertain graph \mathcal{G} represents a probability space consisting of 2^m graphs, sometimes called *possible worlds*, derived by sampling each edge $e \in E$ independently with probability $p(e)$. For $H = (V, E' \subseteq E)$, the event of sampling H as a possible world, denoted $H \sqsubseteq \mathcal{G}$, occurs with probability $\Pr(H \sqsubseteq \mathcal{G}) = \prod_{e \in E'} p(e) \prod_{e \in E \setminus E'} (1 - p(e))$. The notion of possible worlds dates back to Leibniz and *possible world semantics (PWS)* is well-studied in the modal logic literature, beginning with the work of Kripke.

Our work focuses on budgeted dominating sets on vertex-weighted uncertain graphs, i.e., the *Probabilistic Budgeted Dominating Set (PBDS)* problem. The input consists of a vertex-weighted uncertain graph $\mathcal{G} = (V, E, p, \omega)$, with a weight function $\omega : V \rightarrow \mathbb{Q}^+$ and an integer budget k . Set $p(vv) = 1$ for every v . For a vertex u and a set $S \subseteq V$, denote by $\Pr(u \sim S) = 1 - \prod_{v \in S} (1 - p(uv))$ the probability that $u \in S$ or u is connected to some vertex in S . For sets $S_1, S_2 \subseteq V$, the expected coverage (or domination) of S_1 by S_2 is defined as $\mathcal{C}(S_1, S_2) = \sum_{v \in S_1} (w(v) \Pr(v \sim S_2))$. The PBDS problem aims to find a set S of size k that maximizes $\mathcal{C}(V, S)$ over the possible worlds. Its decision version is defined as follows.

Probabilistic budgeted dominating set (PBDS)

Input: A vertex-weighted uncertain graph $\mathcal{G} = (V, E, p, \omega)$, an integer k and a target domination value t .

Question: Is there a set $S \subseteq V$ of size at most k such that $\mathcal{C}(V, S) \geq t$?

Our Results and Discussion. The budgeted dominating set problem is known to have a polynomial time solution on trees. A natural question is if the same applies to the probabilistic version of the problem. We answer this question negatively, showing the following.

► **Theorem 1.** *The PBDS problem is NP-hard on uncertain trees of diameter 4. Furthermore, (i) the PBDS problem on uncertain trees is W[1]-hard for the parameter k , and (ii) an $n^{o(k)}$ time solution to PBDS will falsify the Exponential time hypothesis.*

In order to prove the theorem, we introduce the following problem.

SUBSET $\Sigma - \Pi$ MAXIMIZATION (k -SPM)

Input: A multiset $\mathcal{A} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ of N pairs of positive rationals, an integer k , and a rational t .

Question: Is there a set $S \subseteq [N]$ of size exactly k satisfying $\sum_{i \in S} x_i - \prod_{i \in S} y_i \geq t$?

To establish the complexity of the k -SPM problem, we present a polynomial time reduction from k -SUM to k -SPM, thereby proving that both k -PBDS and k -SPM are NP-hard. Moreover, Downey and Fellows [23] showed that the k -SUM problem is W[1]-hard, implying that if k -SUM has an FPT solution with parameter k , then the W hierarchy collapses. This provides our second hardness result.

► **Theorem 2.** *The k -SPM problem is W[1]-hard for the parameter k . Furthermore, any $N^{o(k)}$ time solution to k -SPM falsify the Exponential time hypothesis.*

The k -SUM problem can be solved easily in $\tilde{O}(n^{\lceil k/2 \rceil})$ time. However, it has been a long-standing open problem to obtain any polynomial improvement over this bound [1, 47]. Patrascu and Williams [48] showed an $n^{o(k)}$ time algorithm for k -SUM falsifies the famous *Exponential time hypothesis* (ETH). Hence, our polynomial time reductions also imply that any algorithm optimally solving k -PBDS or k -SPM must require $n^{\Omega(k)}$ time unless ETH fails.

► **Theorem 3.** *Under the k -SUM conjecture, for any $\varepsilon > 0$, there does not exist an $n^{\lceil k/2 \rceil - \varepsilon}$ time algorithm to PBDS problem on vertex-weighted uncertain trees.*

An intriguing question is whether the k -SPM is substantially harder than k -SUM. For the simple scenario of $k = 2$, the 2-SUM problem has an $O(n \log n)$ time solution. However, it is not immediately clear whether the 2-SPM problem has a truly sub-quadratic time solution (i.e., $O(n^{2-\varepsilon})$ time for some $\varepsilon > 0$). We leave this as an open question. This is especially of interest due to the following result.

► **Theorem 4.** *Let $1 \leq c < 2$ be the smallest real such that 2-SPM problem has an $\tilde{O}(n^c)$ time algorithm. Then, there exists an $\tilde{O}((dn)^{c \lceil k/2 \rceil + 1})$ time algorithm for optimally solving k -PBDS on trees with arbitrary edge-probabilities, for some constant $d > 0$.*

Given the hardness of k -PBDS on uncertain trees, it is of interest to develop efficient approximation algorithms. Clearly, the expected neighborhood size of a vertex set is a submodular function, and thus it is known that the greedy algorithm yields a $(1 - 1/e)$ -approximation for the PBDS problem in general uncertain graphs [39, 46]. For uncertain trees, we improve this by presenting a fully polynomial-time approximation scheme for PBDS.

► **Theorem 5.** *For any integer k , and any n -vertex tree with arbitrary edge probabilities, a $(1 - \epsilon)$ -approximate solution to the optimal probabilistic budgeted dominating set (PBDS) of size k can be computed in time $\tilde{O}(k^2 \epsilon^{-1} n^2)$.*

We also consider a special case that the number of distinct probability edges on the input uncertain tree is bounded above by some constant γ .

► **Theorem 6.** *For any integer k , and an n -vertex tree \mathcal{T} with at most γ edge probabilities, an optimal solution for the PBDS problem on \mathcal{T} can be computed in time $\tilde{O}(k^{(\gamma+2)}n)$.*

We investigate the complexity of PBDS on bounded treewidth graphs. The hardness construction on bounded treewidth graphs is much more challenging. Due to this inherent difficulty, we focus on the *uniform* scenario, where all edge probabilities $p(e)$ are identical. We refer to this version of the problem as *Uni-PBDS*. We show that for any $0 < q < 1$, the Uni-PBDS problem with edge-probability q is $W[1]$ -hard for the pathwidth parameter of the input uncertain graph \mathcal{G} . In contrast, the BDS problem (when all probabilities are one) is FPT when parameterized by the pathwidth of the input graph.

► **Theorem 7.** *Uni-PBDS is $W[1]$ -hard w.r.t. the pathwidth of the input uncertain graph.*

Then, we consider the Uni-PBDS problem with combined k and treewidth parameters. We show that the Uni-PBDS problem can be formulated as a variant of the Extended Monadic Second order (EMS) problem due to Arnborg et al. [6], to derive an FPT algorithm for the Uni-PBDS problem parameterized by the treewidth of \mathcal{G} and k .

► **Theorem 8.** *For any integer k , and any n -vertex uncertain graph of treewidth w with uniform edge probabilities, k -Uni-PBDS can be solved in time $O(f(k, w)n^2)$, and thus is FPT in the combined parameter involving k and w . By dynamic programming on a nice tree decomposition we can show that $f(k, w)$ is $k^{O(w)}$ (we do not present this here).*

Finally, using the structural property of dominating sets from Fomin et al. [28], we derive FPT algorithms parameterized by the budget k in apex-minor-free graphs and planar graphs.

► **Theorem 9.** *For any integer k , and any n -vertex weighted planar or apex-minor free graph, the Uni-PBDS problem can be solved in time $2^{O(\sqrt{k} \log k)} n^{O(1)}$.*

Related Work. Uncertain graphs have been used in the literature to model the uncertainty among relationships in protein-protein interaction networks in bioinformatics [7], road networks [8, 37] and social networks [22, 39, 52, 54]. Connectivity [9, 10, 33, 38, 51, 53], network flows [27, 31], structural-context similarity [55], minimum spanning trees [26], coverage [16, 34, 35, 44, 43], and community detection [11, 49] are well-studied problems on uncertain graphs. In particular, budgeted coverage problems model a wide variety of interesting combinatorial optimization problems on uncertain graphs. For example, the classical facility location problem [36, 40] is a variant of coverage. As another example, in a classical work, Kempe, Kleinberg, and Tardos [39] study influence maximization problem as an expected coverage maximization problem in uncertain graphs. They consider the scenario where influence propagates probabilistically along relationships, under different influence propagation models, like the *Independent Cascade (IC)* and *Linear Threshold (LT)* models, and show that choosing k influencers to maximize the expected influence is NP-hard in the IC model. The coverage problem in the presence of uncertainty was studied extensively also in sensor placement and w.r.t. the placement of light sources in computer vision. SS A special case of the budgeted coverage problem is the *Most Reliable Source (MRS)* problem, where given an uncertain graph $\mathcal{G} = (V, E, p)$, the goal is to find a vertex $u \in V$ such that the expected number of vertices in u 's connected component is maximized. To the best of our knowledge, the computational complexity of MRS is not known, but

it is polynomial time solvable on some specific graph classes like trees and series-parallel graphs [12, 19, 20, 21, 43]. Domination is another special kind of coverage and its complexity is very well-studied. The classical dominating set (DS) problem is known to be $W[2]$ -hard in general graphs [23], and on planar graphs it is fixed parameter tractable with respect to the size of the dominating-set as the parameter [32]. Further, on H -minor-free graphs, the dominating-set problem is solvable in subexponential time [4, 17]. It also admits a linear kernel on H -minor-free graphs and graphs of bounded expansion [3, 25, 29, 30, 50]. On graphs of treewidth bounded by w , the classical dynamic programming approach [15] can be applied to show that the DS problem is FPT when parameterized by w . The *Budgeted Dominating Set (BDS)* problem is known to be NP-hard [42] as well as $W[1]$ -hard for the budget parameter [23]. Furthermore, a subexponential parameterized algorithm is known for BDS on apex-minor-free graphs [28]. The treewidth-parameterized FPT algorithm for the dominating-set problem can be adapted to solve the BDS problem in time $O(3^{wn})$. In particular, for trees there exists a linear running time algorithm. PBDS was studied as MAX-EXP-COVER-1-RF in the survey paper [45], and given a dynamic programming algorithm on a nice tree decomposition with runtime $2^{O(w \cdot \Delta)} n^{O(1)}$, where Δ is the maximum degree of G . The question whether PBDS has a treewidth parameterized FPT algorithm remained unresolved; it is settled in the negative in this work.

2 Preliminaries

Consider a simple undirected graph $G = (V, E)$ with vertex set V and edge set E , and let $n = |V|$ and $m = |E|$. Given a vertex subset $S \subseteq V$, the subgraph induced by S is denoted by $G[S]$. For a vertex $v \in V$, $N(v)$ denotes the set of neighbors of v and $N[v] = N(v) \cup \{v\}$ is the closed neighborhood of v . Let $\deg(v)$ denote the degree of the vertex v in G . A vertex subset $S \subseteq V$ is said to be a *dominating set* of G if every vertex $u \in V \setminus S$ has a neighbor $v \in S$. For an integer $r > 0$, a vertex subset $S \subseteq V$ is said to be an *r -dominating set* of G if for every vertex $u \in V \setminus S$ there exists a vertex $v \in S$ at distance at most r from u . A graph H is said to be an *apex* if it can be made planar by the removal of at most one vertex. A graph G is said to be *apex-minor-free* if it does not contain as its minor some fixed apex graph H . All planar graphs are apex-minor-free as they do not contain as minor the apex graphs $K_{3,3}$ and K_5 . The notations \mathbb{R} , \mathbb{Q} and \mathbb{N} denote, respectively, the sets of real, rational, and natural numbers (including 0). For integers $a \leq b$, define $[a, b]$ to be the set $\{a, a + 1, \dots, b\}$, and for $b > 0$ let $[b] \equiv [1, b]$. Other than this, we follow standard graph theoretic and parameterized complexity terminology [15, 18, 24].

Tree Decomposition. A *Tree decomposition* of an undirected graph $G = (V, E)$ is a pair (T, X) , where T is a tree whose set of nodes is $X = \{X_i \subseteq V \mid i \in V(T)\}$, such that

1. for each edge $uv \in E$, there is an $i \in V(T)$ such that $u, v \in X_i$,
2. for each edge $uv \in E$, there is an $i \in V(T)$ such that $u, v \in X_i$, and
3. for each vertex $v \in V$ the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of T .

The *width* of a tree decomposition (T, X) equals $\max_{i \in V(T)} |X_i| - 1$. The treewidth of a graph G is the minimum width over all tree decompositions of G .

A tree decomposition (T, X) is *nice* if T is rooted by a node r with $X_r = \emptyset$ and every node in T is either an *insert node*, *forget node*, *join node* or *leaf node*. Thereby, a node $i \in V(T)$ is an insert node if i has exactly one child j such that $X_i = X_j \cup \{v\}$ for some $v \notin X_j$; it is a forget node if i has exactly one child j such that $X_i = X_j \setminus \{v\}$ for some $v \in X_j$; it is a join node if i has exactly two children j and h such that $X_i = X_j = X_h$; and it is a leaf node if $X_i = \emptyset$. Given a tree decomposition of width w , a nice tree decomposition of width w and $O(wn)$ nodes can be obtained in linear time [41].

A tree decomposition $(\mathcal{T}, \mathcal{X})$ is said to be a *path decomposition* if \mathcal{T} is a path. The *pathwidth* of a graph G is minimum width over all possible path decompositions of G . Let $\text{pw}(G)$ and $\text{tw}(G)$ denote the pathwidth and treewidth of the graph G , respectively. The pathwidth of a graph G is one lesser than the minimum clique number of an interval supergraph H which contains G as an induced subgraph. It is well-known that the maximal cliques of an interval graph can be linearly ordered so that for each vertex, the maximal cliques containing it occur consecutively in the linear order. This gives a path decomposition of the interval graph. A path decomposition of the graph G is the path decomposition of the interval supergraph H which contains G as an induced subgraph. In our proofs we start with the path decomposition of an interval graph and then reason about the path decomposition of graphs that are constructed from it.

Uncertain Graphs. For an uncertain graph $\mathcal{G} = (V, E, p, \omega)$, the vertex weights and the probabilities associated with the edges are given as rationals. The treewidth and diameter of \mathcal{G} are the same as the treewidth and diameter of its maximal possible world, $G = (V, E)$.

Numerical Approximation. When analyzing our polynomial reductions, we employ numerical analysis techniques to bound the error in numbers obtained as products of an exponential and the root of an integer. We use the following well-known bound on the error in approximating an exponential function by the sum of the lower degree terms in the series expansion.

► **Lemma 10** ([5]). For $z \in [-1, 1]$, e^z can be approximated using the Lagrange remainder as

$$e^z = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots + \frac{z^Q}{Q!} + R_Q(z)$$

where $|R_Q(z)| \leq e/(Q+1)! \leq 1/2^Q$.

We use the following lemma for bounding the error in multiplying approximate values.

► **Lemma 11.** For any set $\{d_1, \dots, d_k\}$ of k reals in the range $[0, 1]$,

$$\prod_{i \in [k]} (1 - d_i) \geq 1 - \sum_{i \in [k]} d_i.$$

Proof. The proof is by induction on k . The base case of $k = 1$ trivially holds. For any two reals $a, b \in [0, 1]$, $(1 - a)(1 - b) \geq 1 - (a + b)$. Applying this result iteratively yields that for any $k \geq 1$, if $\prod_{i \in [k-1]} (1 - d_i) \geq 1 - (\sum_{i \in [k-1]} d_i)$, then

$$\prod_{i \in [k]} (1 - d_i) \geq \left(1 - \left(\sum_{i \in [k-1]} d_i\right)\right)(1 - d_k) \geq 1 - \sum_{i \in [k]} d_i.$$

The claim follows. ◀

3 Hardness Results on Trees

3.1 k -SPM hardness

We first show that the k -SUBSET $\Sigma - \Pi$ MAXIMIZATION (k -SPM) problem is NP-hard by a reduction from the k -SUM problem. Let $\langle X, k \rangle$ with $X = \{x_1, \dots, x_N\}$ be an instance of the k -SUM problem. Let $L = 1 + \max_{i \in [N]} |x_i|$.

Denote by $\langle \mathcal{A}, k, t \rangle$ an instance of the k -SPM problem. Given an instance $\langle X, k \rangle$ of k -SUM, we compute the array $\mathcal{A}(X) = \{(\tilde{x}_i, \tilde{y}_i) \mid i \in [N]\}$ of the k -SPM problem as follows. For $1 \leq i \leq N$, set $\tilde{x}_i := (L + x_i)/(kL)$.

Let $Q = 3 \log_2(kL)$. For $i \in [N]$, define $y_i = e^{x_i/(kL)}$, and let \tilde{y}_i be a rational approximation of y_i that is computed using Lemma 10 such that $0 \leq y_i - \tilde{y}_i \leq 1/2^Q$. The new instance of the k -SPM problem is $\langle \mathcal{A}(X), k, t = 0 \rangle$.

Observe that for each $i \in [N]$, $\tilde{y}_i \geq y_i - 1/2^Q \geq e^{-1/k} - 1/(kL)^3 \geq 1/2$, for $k \geq 3$. Thus, the elements of $\mathcal{A}(X)$ are positive rationals. The next lemma provides a crucial property of any set S of vertices of size k .

► **Lemma 12.** *Let $\lambda = \frac{1}{(2kL)^2}$. For each $S \subseteq [N]$ of size k , $0 \leq \prod_{i \in S} y_i - \prod_{i \in S} \tilde{y}_i \leq \lambda$.*

Proof. Let $\alpha = 1/(kL)^3$. We have:

$$\begin{aligned} \prod_{i \in S} y_i - \prod_{i \in S} \tilde{y}_i &\leq \prod_{i \in S} y_i - \prod_{i \in S} (y_i - \alpha) = \prod_{i \in S} y_i \left(1 - \prod_{i \in S} \left(1 - \frac{\alpha}{y_i}\right)\right) \\ &\leq \prod_{i \in S} y_i \left(\sum_{i \in S} \frac{\alpha}{y_i}\right) \leq e^{\sum_{i \in S} x_i/(kL)} \alpha k e^{1/k} \leq \alpha k e^2 \leq \frac{1}{4(kL)^2}, \end{aligned}$$

where the second inequality is obtained by Lemma 11. The claim follows. ◀

We now establish the correctness of the reduction.

► **Theorem 13.** *The k -SUM problem is polynomial-time reducible to k -SPM.*

Proof. Let $M = \sum_{i \in [N]} (x_i + L)$. Define a real valued function $F(z) = z - e^{-1+z}$ with domain $[0, M/(kL)]$. Observe that $F(1) = 0$ and the derivative is $F'(1) = 0$. The function $F(\cdot)$ is clearly concave, which indicates that:

- (i) $F(z) \leq 0$, for each $z \in [0, M/(kL)]$,
- (ii) $F(z)$ obtains its unique maximum at $z = 1$, where its value is 0, and
- (iii) When restricted to the values in the set $\{z/(kL) \mid z \in [0, M] \text{ is an integer}\}$, $F(z)$ obtains its second largest value at $z = 1 - 1/(kL)$.

For any $S \subseteq [N]$, denote $z_S = \sum_{i \in S} \tilde{x}_i$. For a set $S \subseteq [N]$ of size k , we have:

$$\sum_{i \in S} \tilde{x}_i - \prod_{i \in S} y_i = z_S - e^{\sum_{i \in S} x_i/(kL)} = z_S - e^{\sum_{i \in S} \tilde{x}_i/(kL) - 1/k} \quad (1)$$

$$= z_S - e^{-1} \cdot e^{z_S/(kL)} = F(z_S). \quad (2)$$

By combining Lemma 12 and Eq. (2), we obtain the following.

$$F(z_S) \leq \sum_{i \in S} \tilde{x}_i - \prod_{i \notin S} \tilde{y}_i \leq F(z_S) + \lambda.$$

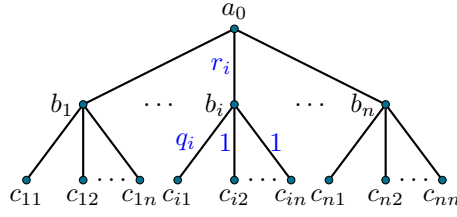
On the other hand, for any set $S \subseteq [N]$ of size k for which $F(z_S) < 0$, we have $F(z_S) \leq F(1 - 1/(kL)) = (1 - 1/(kL)) - e^{-1/(kL)}$.

$$F(z_S) \leq F(1 - 1/(kL)) = (1 - 1/(kL)) - e^{-1/(kL)}.$$

Further,

$$1 - \frac{1}{kL} - e^{-1/(kL)} \leq \left(1 - \frac{1}{kL}\right) - \left(1 - \frac{1}{kL} + \frac{1}{2(kL)^2} - \frac{1}{6(kL)^3}\right) \leq -\frac{1}{4(kL)^2} = -\lambda.$$

So, for a set S , $\sum_{i \in S} \tilde{x}_i - \prod_{i \in S} \tilde{y}_i \geq 0$ if and only if $\sum_{i \in S} \tilde{x}_i = 1$, or equivalently $\sum_{i \in S} x_i = 0$. It follows that $\langle X, k \rangle$ is a yes instance of the k -SUM problem if and only if $\langle \mathcal{A}(X), k, t = 0 \rangle$ is a yes instance of the k -SPM problem. The time to compute \tilde{x}_i and \tilde{y}_i from x_i is polynomial in $Q \cdot \log_2(kL)$, for $1 \leq i \leq N$. Thus, the time-complexity of our reduction is $N \cdot \log_2^{O(1)}(kL)$, which is at most polynomial in N as long as $L = 2^{O(N)}$. Hence, the k -SUM problem is polynomial-time reducible to the k -SPM problem. ◀



■ **Figure 1** Illustration of the lower bound of Theorem 1. Here $p_i = 1 - y_i/(X_{\max} \cdot Y_{\max})$ and $q_i = x_i/(X_{\max} \cdot Y_{\max})^k$ for $i \in [N]$.

Proof of Theorem 2. The reduction given in the proof of Theorem 13 is a parameter preserving reduction for the parameter k . That is, the parameters in the instances of the k -SUM and the k -SPM problem are same in values and the constructed instance of the k -SPM problem is of size polynomial in the input size of the k -SUM instance. Thus, the reduction preserves the parameter k . Since the k -SUM problem is known to be $W[1]$ -hard for the parameter k [2, 23], the k -SPM problem is also $W[1]$ -hard for the parameter k . Further, it is known that under the Exponential time hypothesis (ETH), there cannot exist an $o(N^k)$ time solution for the k -SUM problem [48], so it follows that under ETH there is no $o(N^k)$ time algorithm for k -SPM as well. ◀

3.2 Hardness of PBDS on Uncertain Trees

In this subsection, we show the hardness results for the PBDS problem on trees, establishing Theorem 1. In order to achieve this, we present a polynomial time reduction from k -SPM to PBDS on unweighted trees.

Proof of Theorem 1. In order to prove our claim, we provide a reduction from k -SPM to PBDS. Given an instance $\langle \mathcal{A} = ((x_1, y_1), \dots, (x_N, y_N)), k, t \rangle$ of the k -SPM problem, where t is a rational, an equivalent instance of the PBDS problem is constructed as follows. Let $n = N^2 + N + 1$. Construct an uncertain tree $\mathcal{T} = (V, E, p)$, where the vertex set V consists of three disjoint sets, namely, $A = \{a_0\}$, $B = \{b_1, \dots, b_N\}$, and $C = \{c_{11}, c_{12}, \dots, c_{NN}\}$ (see Figure 1). Note that the uncertain tree \mathcal{T} is considered to be unweighted or unit weight on the vertices. The vertex a_0 is connected by edges to the vertices in B . For each $1 \leq i \leq n$, the vertex b_i is connected by edges to the vertices $c_{i1} \dots, c_{iN}$. Let $X_{\max} = \max\{1, x_1, x_2, \dots, x_N\}$ and $Y_{\max} = \max\{1, y_1, y_2, \dots, y_N\}$. To complete the construction, define the probability function $p : E \rightarrow [0, 1]$ as follows:

$$p(v\bar{v}) = \begin{cases} r_i = 1 - (y_i)/(X_{\max} \cdot Y_{\max}), & \text{if } v\bar{v} = a_0b_i \text{ for } 1 \leq i \leq N, \\ q_i = x_i/(X_{\max} \cdot Y_{\max})^k, & \text{if } v\bar{v} = b_i c_{i1} \text{ for } 1 \leq i \leq N, \\ 1, & \text{otherwise.} \end{cases}$$

Since $x_i, y_i \geq 0$ for each $1 \leq i \leq n$, we have that $p(v, \bar{v}) \in [0, 1]$ is rational for every $(v, \bar{v}) \in E$. This completes the construction of the instance for the PBDS problem. We show that the given instance $\langle \mathcal{A}, k, t \rangle$ is a yes instance of k -SPM if and only if \mathcal{T} has a set S of size k such that $\mathcal{C}(V, S) \geq 1 + (N - 1)k + t/(X_{\max} Y_{\max})^k$.

Let S_{opt} be a set of size k maximizing $\mathcal{C}(V, S_{opt})$ in \mathcal{T} . We show $S_{opt} \subseteq B$. Assume, to the contrary, that there exists some $z \in S_{opt}$ satisfying $z \notin B$. Consider $i \in [N]$ such that none of the vertices c_{i1}, \dots, c_{iN} lie in S_{opt} . Such i must exist since $|S_{opt}| = k$. If $z \in C$, then replacing z by b_i results in a set $S' = S_{opt} \setminus \{z\} \cup \{b_i\}$ such that $\mathcal{C}(V, S') \geq \mathcal{C}(V, S_{opt}) + N - 3$,

contradicting the optimality of S_{opt} . Hence, S_{opt} must be contained in $A \cup B$. In this case, z must be a_0 . Now the set $S' = S_{opt} \setminus \{z\} \cup \{b_i\}$ is such that $\mathcal{C}(V, S') \geq \mathcal{C}(V, S_{opt}) + N/2 - 2$. Since $N \geq 6$, this contradicts the optimality of S_{opt} . It follows that $S_{opt} \subseteq B$.

Next, consider a set $S \subseteq B$ of size k , and let $I_S = \{i \in [N] \mid b_i \in S\}$. We have

$$\begin{aligned} \mathcal{C}(V, S) &= \left(1 - \prod_{i \in I_S} (1 - p(a_0, b_i))\right) + \left(\sum_{i \in I_S} (p(b_i, c_{i1}) + N - 1)\right) \\ &= 1 + (N - 1)k + \sum_{i \in I_S} x_i / (X_{\max} \cdot Y_{\max})^k - \prod_{i \in I_S} y_i / (X_{\max} \cdot Y_{\max}) \\ &= 1 + (N - 1)k + (X_{\max} \cdot Y_{\max})^{-k} \left(\sum_{i \in I_S} x_i - \prod_{i \in I_S} y_i\right). \end{aligned}$$

This formulation of the coverage function shows that the given instance $\langle \mathcal{A}, k, t \rangle$ is a yes instance of k -SPM if and only if \mathcal{T} has a set S of vertices of size k such that $\mathcal{C}(V, S) \geq 1 + (N - 1)k + t / (X_{\max} Y_{\max})^k$. Thus, the k -SPM problem is reduced in polynomial time to PBDS on unweighed trees.

It remains to prove NP-hardness, W[1]-hardness, and $n^{o(k)}$ lower-bound under ETH. Note that the above reduction is a parameterized preserving reduction for the parameter k . That is, the parameter k in the k -SPM problem is the solution size (also called k) parameter for the PBDS problem. Since the k -SPM problem (i) is NP-hard, (ii) is W[1]-hard for the parameter k , and (iii) cannot have time complexity $n^{o(k)}$ under the Exponential time hypothesis (by Theorem 2), it follows that the same hardness results hold for PBDS as well. Therefore, the PBDS problem on uncertain trees (i) is NP-hard, (ii) is W[1]-hard for the parameter k , and (iii) cannot have time complexity $n^{o(k)}$ if ETH holds true. ◀

The k -SUM conjecture [1, 47] states that the k -SUM, for the parameters N and k , requires at least $N^{\lceil k/2 \rceil - o(1)}$ time.

► **Conjecture 14** (*k -SUM Conjecture*). *There do not exist a $k \geq 2$, an $\varepsilon > 0$, and an algorithm that succeeds (with high probability) in solving k -SUM in $N^{\lceil k/2 \rceil - \varepsilon}$ time.*

Proof of Theorem 3. Consider the uncertain tree \mathcal{T} constructed in the proof of Theorem 1. We set $n_0 = 0$. Modify the original construction of \mathcal{T} by deleting the $N - 2$ vertices: $c_{i3}, c_{i4}, \dots, c_{iN}$, and setting $\omega_{c_{i2}} = N - 1$, for $1 \leq i \leq N$. Thus the tree contains exactly $n = 3N + 1$ vertices. Now, k -SUM is reducible to k -SPM, and k -SPM is reducible to PBDS, both in polynomial time, and, moreover the parameter k remains unaltered and the size of problem grows by at most constant factor. This shows that, for $\varepsilon > 0$, an $n^{\lceil k/2 \rceil - \varepsilon}$ time algorithm to weighted PBDS implies an $N^{\lceil k/2 \rceil - \Omega(\varepsilon)}$ time algorithm to k -SUM, thereby, falsifying the k -SUM conjecture. ◀

Note that since the PBDS problem is NP-hard on trees, it is also para-NP-hard [15, 24] for the treewidth parameter.

4 Hardness of Uni-PBDS for the pathwidth parameter

In this section, we show that even for the restricted case of uniform probabilities, the Uni-PBDS problem is W[1]-hard for the pathwidth parameter, and thus also for treewidth (Theorem 7). This is shown by a reduction from the MULTI-COLORED CLIQUE problem to the Uni-PBDS problem. It is well-known that the MULTI-COLORED CLIQUE problem is W[1]-hard for the parameter solution size [23].

MULTI-COLORED CLIQUE

Input: A positive integer k and a k -colored graph G .

Parameter: k

Question: Does there exist a clique of size k with one vertex from each color class?

Let $(G = (V, E), k)$ be an input instance of the MULTI-COLORED CLIQUE problem, with n vertices and m edges. Let $V = (V_1, \dots, V_k)$ denote the partition of the vertex set V in the input instance. We assume, without loss of generality, $|V_i| = n$ for each $i \in [k]$. For each $1 \leq i \leq k$, let $V_i = \{u_{i,\ell} \mid 1 \leq \ell \leq n\}$.

4.1 Gadget based reduction from Multi-Colored Clique

Let (G, k) be an instance of the MULTI-COLORED CLIQUE problem. For any probability $0 < p < 1$, and for any integer f such that $f > \max\{knm, n + k^2/p\}$, our reduction constructs an uncertain graph \mathcal{G} . The output of the reduction is an instance (\mathcal{G}, k', t') of the Uni-PBDS problem where each edge has probability p , $k' = (n + 1)(m + kn)$ and $t' = (kn + m)((n + 1)fp + n + np + 1 + 2(1 - (1 - p)^n)) + 4\binom{k}{2}(1 - (1 - p)^{n+1})$. In the presentation below, we show that this choice of k' and t' ensures that there is a set of size k' with expected domination at least t' in \mathcal{G} if and only if G has a multi-colored clique of size k .

We first construct a gadget graph to represent the vertices and edges of the input instance of the MULTI-COLORED CLIQUE problem. We construct two types of gadgets, \mathcal{D} and \mathcal{I} in the reduction, illustrated in Figure 2 (in Appendix A.1). The gadget \mathcal{I} is the primary gadget and \mathcal{D} is a secondary gadget used to construct \mathcal{I} . When we refer to a gadget, we mean the primary gadget \mathcal{I} unless the gadget \mathcal{D} is specified. For each vertex and edge in the given graph, our reduction has a corresponding gadget. The gadget \mathcal{D} is defined as follows.

Gadget of type \mathcal{D} . Given a pair of vertices u and v , the gadget $\mathcal{D}_{u,v}$ consists of vertices u , v , and f additional vertices. The vertices u and v are made adjacent to every other vertex. We refer to the vertices u and v as *heads*, and remaining vertices of $\mathcal{D}_{u,v}$ as *tails*, and u and v are said to be connected by the gadget $\mathcal{D}_{u,v}$.

► **Observation 15.** *The pathwidth of a gadget of type \mathcal{D} is 2.*

Gadget of type \mathcal{I} . We begin the construction of the gadget with $2n$ vertices partitioned into two sets where each partition contains n vertices. Let $A = \{a_1, \dots, a_n\}$ and $C = \{c_1, \dots, c_n\}$ be this partition. For each $i \in [n]$, vertices a_i and c_i are connected by the gadget \mathcal{D}_{a_i, c_i} . Let h_a and h_c be two additional vertices connected by the gadget \mathcal{D}_{h_a, h_c} . The vertices in the sets A and C are made adjacent to h_a and h_c , respectively. This completes the construction of the gadget. In the reduction, a gadget of type \mathcal{I} is denoted by the symbol \mathcal{I} along with an appropriate subscript based on whether the gadget is associated with a vertex or an edge.

▷ **Claim 16.** *The pathwidth of a gadget of type \mathcal{I} is at most 4.*

Description of the reduction. For $1 \leq i < j \leq k$, let $E_{i,j} = \{xy \mid x \in V_i, y \in V_j\}$ be the set of edges with one end point in V_i and the other in V_j in G . For each $1 \leq i < j \leq k$, the graph \mathcal{G} has an induced subgraph \mathcal{G}_i corresponding to V_i , and has an induced subgraph $\mathcal{G}_{i,j}$ for the edge set $E_{i,j}$. We refer to \mathcal{G}_i as a vertex-partition block and $\mathcal{G}_{i,j}$ as an edge-partition block. Inside block \mathcal{G}_i , there is a gadget of type \mathcal{I} for each vertex in V_i , and in the block $\mathcal{G}_{i,j}$, there is a gadget for each edge in $E_{i,j}$. For a vertex $u_{i,x}$, \mathcal{I}_x denotes the gadget corresponding to $u_{i,x}$ in the partition V_i , and for an edge e , \mathcal{I}_e denotes the gadget corresponding to e . The blocks are appropriately connected by connector vertices which are defined below.

We start by defining the structure of a block denoted by B . The definition of the block applies to both the vertex-partition block and the edge-partition block. A block B consists of gadgets and additional vertices as follows (See Figure 3 in Appendix A.1).

- The block B corresponding to the vertex-partition block \mathcal{G}_i for any $i \in [k]$ is described as follows: for each $\ell \in [n]$, add a gadget \mathcal{I}_ℓ to the vertex-partition block \mathcal{G}_i , to represent the vertex $u_{i,\ell} \in V_i$. In addition to the gadgets, we add $n + 1$ vertices to the block B described as follows: Let $F(B) = \{b_1, \dots, b_n, d_i\}$ be the set of additional vertices that are added to the block B . For each $\ell \in [n]$, the vertices in the set C of the gadget \mathcal{I}_ℓ in the block B are made adjacent to b_ℓ . For each $\ell \in [n]$, the vertices in the set A of the gadget \mathcal{I}_ℓ in the block B are made adjacent to d_i .
- The block B corresponding to the edge-partition block $\mathcal{G}_{i,j}$ for any $1 \leq i < j \leq k$ is described as follows: for each $e \in E_{i,j}$, add a gadget \mathcal{I}_e in the edge-partition block $\mathcal{G}_{i,j}$, to represent the edge e . In addition to the gadgets, we add $|E_{i,j}| + 1$ vertices to the block B described as follows: Let $F(B) = \{b_e \mid e \in E_{i,j}\} \cup \{d_{i,j}\}$ be the set of additional vertices that are added to the block B . For each $e \in E_{i,j}$, the vertices in the set C of the gadget \mathcal{I}_e in the block B are made adjacent to b_e . For each $e \in E_{i,j}$, the vertices in the set A of the gadget \mathcal{I}_e in the block B are made adjacent to $d_{i,j}$.

The blocks defined above are connected by the connector vertices described next. These connector vertices are used to connect the edge-partition blocks and vertex-partition blocks, and thus ensure that each edge in G is appropriately represented in \mathcal{G} . Let $R = \{r_{i,j}^i, s_{i,j}^i, r_{i,j}^j, s_{i,j}^j \mid 1 \leq i < j \leq k\}$ be the connector vertices. The blocks are connected based on the cases described below. The connections involving the \mathcal{I} gadgets in two vertex-partition blocks and an \mathcal{I} gadget in an edge-partition block is illustrated in Figure 4 in Appendix A.1. First, we describe the connection of vertex-partition blocks corresponding V_i and V_j to the appropriate connector vertices. Following this, we describe the connection of the two vertex-partition blocks to the edge-partition block corresponding to $E_{i,j}$ through the appropriate connector vertices.

For each $i \in [k]$, each $i < j \leq k$ and each $\ell \in [n]$,

- for each $1 \leq t \leq \ell$, a_t in the gadget \mathcal{I}_ℓ of \mathcal{G}_i is made adjacent to $s_{i,j}^i$, and
- for each $\ell \leq t \leq n$, a_t in the gadget \mathcal{I}_ℓ of \mathcal{G}_i is made adjacent to the vertex $r_{i,j}^i$.

For each $i \in [k]$, each $1 \leq j < i$ and each $\ell \in [n]$,

- for each $1 \leq t \leq \ell$, a_t in the gadget \mathcal{I}_ℓ of \mathcal{G}_i is made adjacent to the vertex $s_{j,i}^i$, and
- for each $\ell \leq t \leq n$, a_t in the gadget \mathcal{I}_ℓ of \mathcal{G}_i is made adjacent to the vertex $r_{j,i}^i$.

Now, we describe the edges to connect the \mathcal{I} gadgets in the vertex-partition blocks \mathcal{G}_i and \mathcal{G}_j and to the appropriate \mathcal{I} gadgets in the edge-partition block $\mathcal{G}_{i,j}$. For each $1 \leq i < j \leq k$, and for each $e = u_{i,x}u_{j,y} \in E_{i,j}$,

- for each $1 \leq t \leq x$, a_t in the gadget \mathcal{I}_e of $\mathcal{G}_{i,j}$ is made adjacent to the vertex $r_{i,j}^i$, and
- for each $x \leq t \leq n$, a_t in the gadget \mathcal{I}_e of $\mathcal{G}_{i,j}$ is made adjacent to the vertex $s_{i,j}^i$.
- for each $1 \leq t \leq y$, a_t in the gadget \mathcal{I}_e of $\mathcal{G}_{i,j}$ is made adjacent to the vertex $r_{i,j}^j$, and
- for each $y \leq t \leq n$, a_t in the gadget \mathcal{I}_e of $\mathcal{G}_{i,j}$ is made adjacent to the vertex $s_{i,j}^j$.

This completes the construction of the graph \mathcal{G} with $O(mn^2)$ vertices and $O(mn^3)$ edges.

▷ **Claim 17.** The pathwidth of a block B is at most 6.

The following lemma bounds the pathwidth of the graph \mathcal{G} by a polynomial in k .

▶ **Lemma 18.** *The pathwidth of the graph \mathcal{G} is at most $4\binom{k}{2} + 6$.*

Due to space constraints, we have deferred the complete proof to the expanded version.

Proof of Theorem 7. Given an instance (G, k) of MULTI-COLORED CLIQUE, the instance (\mathcal{G}, k') is constructed in polynomial time where k' and t' are polynomial in input size. By Lemma 18, the pathwidth of \mathcal{G} is a quadratic function of k . Finally, we can also show that the Uni-PBDS instance (\mathcal{G}, k', t') output by the reduction is equivalent to the MULTI-COLORED CLIQUE instance (G, k) that was input to the reduction. Since MULTI-COLORED CLIQUE is known to be $W[1]$ -hard for the parameter k , it follows that the Uni-PBDS problem is $W[1]$ -hard with respect to the pathwidth parameter of the input graph. ◀

This completes the proof of Theorem 7.

5 PBDS on Trees: PTAS and Exact Algorithm

In this section, we present our algorithmic results for the PBDS problem on trees. Throughout this section, assume \mathcal{T} is rooted at some vertex r . For each $x \in V$, denote by $\text{PAR}(x)$ the parent of x in V , and by $\mathcal{T}(x)$ the subtree of \mathcal{T} rooted at x .

5.1 PTAS for PBDS on Trees

For each $v \in V$ and each $b \in [0, k]$, define $\mathcal{Y}_v(\text{par}, \text{curr}, b)$ to be the optimal value of $\mathcal{C}(V(\mathcal{T}(v)), S)$ where par and curr are boolean indicator variables that, respectively, denote whether or not $\text{PAR}(v)$ and v are in S (written below as $I_{\text{PAR}(v) \in S}$ and $I_{v \in S}$), and b denotes the number of descendants of v in S . Formally, $\mathcal{Y}_v(\text{par}, \text{curr}, b)$ is represented as follows:

$$\arg \max \left\{ \sum_{x \in \mathcal{T}(v)} \mathcal{C}(x, S) \mid S \subseteq V, |S \cap (\mathcal{T}(v) \setminus v)| = b, \text{curr} = I_{v \in S}, \text{par} = I_{\text{PAR}(v) \in S} \right\}$$

The main idea behind our PTAS is to use the rounding method. Instead of computing \mathcal{Y}_x , we compute its approximation, represented as $\widehat{\mathcal{Y}}_x$. This is done in a bottom-up fashion, starting from leaf nodes of \mathcal{T} . For each $x \in V$, define $\delta(x)$ to be $|\mathcal{Y}_x - \widehat{\mathcal{Y}}_x|$. Throughout our algorithm, we maintain the invariant that $\widehat{\mathcal{Y}}_x \leq \mathcal{Y}_x$, for every $x \in V$.

We now present an algorithm to compute $\widehat{\mathcal{Y}}$. Since \mathcal{Y}_x is easy to compute for a leaf x , we set $\widehat{\mathcal{Y}}_x = \mathcal{Y}_x$. For a leaf x , $\mathcal{Y}_x(\text{par}, \text{curr}, b)$ is (i) undefined if $b \neq 0$, (ii) ω_x if $\text{curr} = 1, b = 0$, (iii) $\omega_x p_{(\text{PAR}(x), x)}$ if $\text{par} = 1, \text{curr} = 0, b = 0$, and (iv) zero otherwise. Consider a non-leaf v . Let z_1, \dots, z_t be v 's children in \mathcal{T} , and z_0 be v 's parent in \mathcal{T} (if exists). Let $\mathcal{L}(\beta)$, for $\beta \geq 0$, denote the collection of all integral vectors $\sigma = (b_1, \text{curr}_1, \dots, b_t, \text{curr}_t)$ of length $2t$ satisfying (i) $\text{curr}_i \in \{0, 1\}$ and $b_i \geq 0$, for $i \in [1, t]$, and (ii) $\sum_{i \in [1, t]} (b_i + \text{curr}_i) = \beta$. In our representation of σ as $(b_1, \text{curr}_1, \dots, b_t, \text{curr}_t)$, the term curr_i corresponds to the indicator variable representing whether or not z_i lies in our tentative set S , and b_i corresponds to the cardinality of $S \cap (V(\mathcal{T}(z_i)) \setminus z_i)$. Further, for $i \in [1, t]$, let $\mathcal{L}_i(\beta)$ be the collection of those vectors $\sigma = (b_1, \text{curr}_1, \dots, b_t, \text{curr}_t) \in \mathcal{L}(\beta)$ that satisfy $b_j, \text{curr}_j = 0$ for $j > i$.

For a given $\text{curr}, \text{par}, b \geq 0$, we now explain the computation of $\widehat{\mathcal{Y}}_v(\text{par}, \text{curr}, b)$. Assume that we have already computed the approximate values $\widehat{\mathcal{Y}}_{z_i}$ ($i \in [1, t]$) corresponding to v 's children in \mathcal{T} . Setting $W = \max_{u \in V} \omega_u$, and using the scaling factor $M = \epsilon W/n$, let

$$A(\sigma) = \begin{cases} \omega_v, & \text{if curr}=1, \\ M \left[\frac{\omega_v}{M} \left(1 - (1 - \text{par} \cdot p_{(z_0, v)}) \cdot \prod_{\substack{i \in [1, t] \\ \text{curr}_i=1}} (1 - p_{(z_i, v)}) \right) \right], & \text{otherwise,} \end{cases} \quad (3)$$

$$B(\sigma) = \sum_{i \in [1, t]} \widehat{\mathcal{Y}}_{z_i}(\text{curr}, \text{curr}_i, b_i), \quad (4)$$

$$\widehat{\mathcal{Y}}_v(\text{par}, \text{curr}, b) = \max_{\sigma \in \mathcal{L}(b)} (A(\sigma) + B(\sigma)). \quad (5)$$

In order to efficiently compute $\widehat{\mathcal{Y}}_v$, we define the notion of *preferable* vectors. For any two vectors $\sigma_1, \sigma_2 \in \mathcal{L}(\beta)$, we say that σ_1 is *preferred* over σ_2 (and write $\sigma_1 \geq \sigma_2$) if both (i) $A(\sigma_1) \geq A(\sigma_2)$, and (ii) $B(\sigma_1) \geq B(\sigma_2)$. For $i \in [1, t]$, let $\mathcal{L}_i^*(\beta)$ be a *maximal* subset of $\mathcal{L}_i(\beta)$ such that $\sigma_1 \not\geq \sigma_2$ for any two vectors $\sigma_1, \sigma_2 \in \mathcal{L}_i^*(\beta)$.

Define $\phi_v = |\{A(\sigma) \mid \sigma \in \mathcal{L}(\beta), \text{ for } \beta \in [0, k]\}|$. The following observation is immediate by the definition of \mathcal{L}_i^* .

► **Observation 19.** For each $i \in [1, t]$ and $\beta \in [0, k]$, $|\mathcal{L}_i^*(\beta)| \leq \phi_v$.

In order to compute $\widehat{\mathcal{Y}}_v(\text{par}, \text{curr}, b)$, we explicitly compute and store $\mathcal{L}_i^*(\beta)$, for $1 \leq i \leq t$. The set $\mathcal{L}_1^*(\beta)$ is quite easy to compute. Let $\sigma_1 = (\beta, 0, 0, \dots, 0)$ and $\sigma_2 = (\beta - 1, 1, 0, \dots, 0)$ be the only two vectors lying in $\mathcal{L}_1(\beta)$. Then $\mathcal{L}_1^*(\beta)$ is that vector among σ_1 and σ_2 that maximizes the sum $A(\sigma) + B(\sigma)$.

The lemma below provides an iterative procedure for computing the sets $\mathcal{L}_i^*(\beta)$, for $i \geq 2$.

► **Lemma 20.** For every $i, \beta \geq 1$, the set $\mathcal{L}_i^*(\beta)$ can be computed from $\mathcal{L}_{i-1}^*(\beta)$ in time $\widetilde{O}(\beta + \sum_{\alpha \in [0, \beta]} |\mathcal{L}_{i-1}^*(\alpha)|)$.

Proof. Initialize $\mathcal{L}_i^*(\beta)$ to \emptyset . At each stage, maintain the list $\mathcal{L}_i^*(\beta)$ sorted by the values $A(\cdot)$, and reverse-sorted by the values $B(\cdot)$. Our algorithm to compute $\mathcal{L}_i^*(\beta)$ involves the following steps.

1. For each $\text{curr} \in \{0, 1\}$ and $b \in [0, \beta]$, first compute a set $\mathcal{P}_{b, \text{curr}}$ obtained by replacing the values b_i and curr_i in each $\sigma \in \mathcal{L}_{i-1}^*(\beta - (\text{curr} + b))$ by b and curr respectively. Let $\mathcal{P} = \bigcup_{b \in [0, \beta], \text{curr} \in \{0, 1\}} \mathcal{P}_{b, \text{curr}}$.
2. For each $\sigma \in \mathcal{P}$, check in $O(\log |\mathcal{P}|)$ time if there is a $\sigma' \in \mathcal{L}_i^*(\beta)$ that is preferred over σ (i.e. $\sigma' \geq \sigma$). If no such σ' exists, then (a) add σ to $\mathcal{L}_i^*(\beta)$, and (b) remove all those σ'' from $\mathcal{L}_i^*(\beta)$ that are less preferred than σ , that is, $\sigma'' < \sigma$.

The runtime of the algorithm is $O(\beta + |\mathcal{P}| \log |\mathcal{P}|)$ which is at most $\widetilde{O}(\beta + \sum_{\alpha \in [0, \beta]} |\mathcal{L}_{i-1}^*(\alpha)|)$.

Next we now prove its correctness. Consider a $\sigma = (b_1, \text{curr}_1, \dots, b_t, \text{curr}_t) \in \mathcal{L}_i(\beta)$. It suffices to show that if $\sigma \notin \mathcal{P}_{b_i, \text{curr}_i}$, then there exists a $\sigma' \in \mathcal{P}_{b_i, \text{curr}_i}$ satisfying $\sigma' \geq \sigma$. Let σ_0 be obtained from σ by replacing b_i, curr_i with 0. Since $\sigma \notin \mathcal{P}_{b_i, \text{curr}_i}$, it follows that $\sigma_0 \notin \mathcal{L}_{i-1}(\beta - (b_i + \text{curr}_i))$. So there must exist a vector $\sigma'_0 = (b'_1, \text{curr}'_1, \dots, b'_t, \text{curr}'_t) \in \mathcal{L}_{i-1}(\beta - (b_i + \text{curr}_i))$ satisfying $A(\sigma'_0) \geq A(\sigma_0)$ and $B(\sigma'_0) \geq B(\sigma_0)$. Let σ' be the vector obtained from σ'_0 by replacing b'_i, curr'_i with b_i, curr_i . It can be easily verified from Eq. (3) and (4), that $A(\sigma') \geq A(\sigma)$ and $B(\sigma') \geq B(\sigma)$. Since the constructed σ' indeed lies in $\mathcal{P}_{b_i, \text{curr}_i}$, the proof follows. ◀

The following claim is an immediate corollary of Lemma 20.

► **Lemma 21.** The value of $\widehat{\mathcal{Y}}_v(\text{par}, \text{curr}, b)$, for any $\text{par}, \text{curr} \in \{0, 1\}$ and $b \in [0, k]$, is computable in $\widetilde{O}(b \cdot \deg(v) \cdot \phi_v)$ time, given the values of $\widehat{\mathcal{Y}}_{z_i}$ for $i \leq t$.

Proof. Observe that $\widehat{\mathcal{Y}}_v(\text{par}, \text{curr}, b) = \max_{\sigma \in \mathcal{L}_i^*(b)} (A(\sigma) + B(\sigma))$, where $A(\sigma)$ and $B(\sigma)$ are as defined in Eq. (3) and (4). By Observation 19 and Lemma 20, the total computation time of the set $\mathcal{L}_i^*(b)$ is at most $\widetilde{O}(b \cdot t \cdot \phi_v)$, which is equal to $\widetilde{O}(b \cdot \deg(v) \cdot \phi_v)$. ◀

Lemma 21 implies that starting from leaf nodes, the values $\widehat{\mathcal{Y}}_x(\text{par}, \text{curr}, b)$ can be computed in bottom-up manner, for each valid choice of triplet $(\text{par}, \text{curr}, b)$ and each $x \in V$, in total time $\widetilde{O}(k^2 n \cdot \max_{v \in V} \phi_v)$. We now prove $\phi_v = O(\epsilon^{-1} n)$. If $\text{curr} = 1$, then $A(\sigma)$ takes only one value. If $\text{curr} = 0$, then the value of $A(\sigma)$ is a multiple of M and is also bounded above by W . This implies that the number of distinct values $A(\sigma)$ can take is indeed bounded by $W/M = O(\epsilon^{-1} n)$.

► **Proposition 22.** *Computing $\widehat{\mathcal{Y}}_x$ for all $x \in V$ takes in total $\widetilde{O}(k^2 n \cdot \max_{x \in V} \phi_x) = \widetilde{O}(k^2 \epsilon^{-1} n^2)$ time.*

5.2 Approximation Analysis of PTAS on Trees

We provide here the approximation analysis of the $(1 - \epsilon)$ -bound. Let

$$\begin{aligned} S_{opt} &= \arg \max_{S \subseteq V, |S|=k} \mathcal{C}(V, S) = \arg \max_{S \subseteq V, |S|=k} \sum_{x \in V} \omega_x \Pr(x \sim S), \\ \widehat{S}_{opt} &= \arg \max_{S \subseteq V, |S|=k} \left(\sum_{x \in S} \omega_x \Pr(x \sim S) + \sum_{x \in V \setminus S} M \left\lfloor \frac{\omega_x \Pr(x \sim S)}{M} \right\rfloor \right). \end{aligned}$$

Observe that $\max\{\mathcal{Y}_r(0, 0, k), \mathcal{Y}_r(0, 1, k-1)\} = \mathcal{C}(V, S_{opt})$ and $\max\{\widehat{\mathcal{Y}}_r(0, 0, k), \widehat{\mathcal{Y}}_r(0, 1, k-1)\} = \mathcal{C}(V, \widehat{S}_{opt})$. The following lemma proves that \widehat{S}_{opt} indeed achieves a $(1 - \epsilon)$ -approximation bound.

► **Lemma 23.** $(1 - \epsilon) \mathcal{C}(V, S_{opt}) \leq \mathcal{C}(V, \widehat{S}_{opt}) \leq \mathcal{C}(V, S_{opt})$.

Proof. In order to prove the first inequality, we first show that $\mathcal{C}(V, S_{opt}) - \mathcal{C}(V, \widehat{S}_{opt}) \leq \epsilon \mathcal{C}(V, S_{opt})$.

$$\begin{aligned} \mathcal{C}(V, S_{opt}) - \mathcal{C}(V, \widehat{S}_{opt}) &\leq \max_{\substack{S \subseteq V \\ |S|=k}} \left(\sum_{x \in V} \omega_x \Pr(x \sim S) - \sum_{x \in S} \omega_x \Pr(x \sim S) - \sum_{x \in V \setminus S} M \left\lfloor \frac{\omega_x \Pr(x \sim S)}{M} \right\rfloor \right) \\ &\leq (n - k)M \leq \epsilon W \leq \epsilon \mathcal{C}(V, S_{opt}). \end{aligned}$$

Next, for each $x \in V$ and $S \subseteq V$, we have $M \lfloor M^{-1} \omega_x \Pr(x \sim S) \rfloor \leq \omega_x \Pr(x \sim S)$, thereby implying that $\mathcal{C}(V, \widehat{S}_{opt}) \leq \mathcal{C}(V, S_{opt})$. This completes our proof. ◀

Proof of Theorem 5. For any integer k , any n -vertex tree \mathcal{T} with arbitrary edge probabilities, and for every $\epsilon > 0$, a $(1 - \epsilon)$ approximate solution can be computed in time $\widetilde{O}(k^2 \epsilon^{-1} n^2)$. This follows from Proposition 22 and Lemma 23. This completes the proof Theorem 5. ◀

5.3 Linear-time algorithm for Uni-PBDS on Trees

We next establish our result for the scenario of Uni-PBDS on trees (Thm. 6). In fact, this result holds for a somewhat broader scenario, wherein, for each vertex x , the cardinality of $\text{PROB}_x = \{p_e \mid e \text{ is incident to } x\}$ is bounded above by some constant γ .

Proof of Theorem 6. Observe that the only place where approximation was used in our PTAS was in bounding the number of distinct values that can be taken by $A(\sigma)$ in Eq. (3). In order to obtain an exact solution for the bounded probabilities setting, the only modification performed in our algorithm is to redefine $A(\sigma)$ as follows.

$$A(\sigma) = \omega_v \cdot I_{(curr=1)} + \omega_v \left(1 - (1 - par \cdot p_{(z_0, v)}) \cdot \prod_{\substack{i \in [1, t] \\ curr_i=1}} (1 - p_{(z_i, v)}) \right) \cdot I_{(curr \neq 1)}.$$

It can be verified that the algorithm correctly computes \mathcal{Y}_x at each step, that is, $\delta(x)$ is essentially zero. The time it takes to compute $\mathcal{Y}_v(par, curr, b)$, for a non-leaf v , crucially depends on the cardinality of $\{A(\sigma) \mid \sigma \in \mathcal{L}_t^*(b)\}$, where t is the number of children of v in \mathcal{T} . Observe that the number of distinct values $A(\sigma)$ can take is at most $b^{|\text{PROB}_x|} = O(k^\gamma)$. This along with Lemma 22 implies that the total runtime of our exact algorithm is $\widetilde{O}(k^{\gamma+2} n)$. ◀

5.4 Solving PBDS optimally on general trees

Let $c \geq 1$ be the smallest real such that 2-SPM problem has an $\tilde{O}(N^c)$ time algorithm. We will show that in such a case, k -PBDS can be solved optimally on trees with arbitrary probabilities in $\tilde{O}((\delta N)^{c \lceil k/2 \rceil + 1})$ time, for a constant $\delta > 0$.

For any node $v \in \mathcal{T}$, let \mathcal{T}_v^i , for $1 \leq i \leq \deg(v)$, represent the components of the subgraph $\mathcal{T} \setminus \{v\}$. We start with the following lemma (proved in Appendix B.1), which is easy to prove using a standard counting argument.

► **Lemma 24.** *For any set S of size k in \mathcal{T} , there exist a node $v \in \mathcal{T}$ and an index $q \in [1, \deg(v)]$ such that the cardinalities of the sets $S \cap (\bigcup_{i \leq q} \mathcal{T}_v^i)$, $S \cap (\mathcal{T}_v^q)$ and $S \cap (\bigcup_{i \geq q} \mathcal{T}_v^i)$ are all bounded by $k/2$.*

For the rest of this section, we refer to a tuple (v, q) satisfying the conditions stated in Lemma 24 as a *valid pair*. Let us suppose we are provided with a valid pair (v, q) . For sake of convenience, we assume that \mathcal{T} is rooted at node v . Let $U_0 = \mathcal{T}_v^q$, $U_1 = \bigcup_{i \leq q} \mathcal{T}_v^i$, and $U_2 = \bigcup_{i \geq q} \mathcal{T}_v^i$. Also let $\{x_1, \dots, x_\alpha\}$ be the children of v in U_1 , where $\alpha = q - 1$; let $\{y_1, \dots, y_\beta\}$ be the children of v in U_2 , where $\beta = \deg(v) - q$; and let z be q^{th} child of v .

An important observation is that if the optimal S contains v , then the problem is easily solvable in $O(k^2 \cdot n^{k/2})$ time, as then the structures U_0 , U_1 and U_2 become independent. Indeed, it suffices to consider all $O(k^2)$ partitions of $k - 1$ into triplet (c_0, c_1, c_2) consisting of integers in the range $[0, k/2]$, and next we iterate over all the $c_i \leq k/2$ subsets in U_i . This takes in total $O(k^2 n^{k/2})$ time. So the challenge is to solve the problem when v is not contained in S . Assuming (v, q) is a valid pair, and v is not contained in the optimal S , the solution S to k -PBDS is the union $S_0 \cup S_1 \cup S_2$ of the tuple $(S_0, S_1, S_2) \in U_0 \times U_1 \times U_2$ that maximizes

$$\mathcal{C}(U_0, S_0) + \mathcal{C}(U_1, S_1) + \mathcal{C}(U_2, S_2) + \omega_v \left(1 - (1 - d \cdot p(v, z)) \cdot \prod_{\substack{i \in [1, \alpha] \\ x_i \in S_1}} (1 - p(v, x_i)) \prod_{\substack{j \in [1, \beta] \\ y_j \in S_2}} (1 - p(v, y_j)) \right) \quad (6)$$

where d is an indicator variable denoting whether or not $z \in S_0$, and $|S_1|, |S_2|, |S_3|$ are integers in the range $[0, k/2]$ that sum up to k .

Define Γ to be set of all quadruples (d, c_0, c_1, c_2) comprising of integers in the range $[0, k/2]$ such that $d \in \{0, 1\}$ and $c_0 + c_1 + c_2 = k$. For each $\gamma = (d, c_0, c_1, c_2) \in \Gamma$, let

$$\begin{aligned} \mathcal{L}_\gamma^1 &= \left\{ \left(\frac{\mathcal{C}(U_1, S_1)}{\omega_v (1 - d \cdot p(v, z))}, \prod_{\substack{i \in [1, \alpha] \\ x_i \in S_1}} (1 - p(v, x_i)) \right) \mid S_1 \subseteq U_1 \text{ is of size } c_1 \right\}, \\ \mathcal{L}_\gamma^2 &= \left\{ \left(\frac{\mathcal{C}(U_2, S_2)}{\omega_v (1 - d \cdot p(v, z))}, \prod_{\substack{j \in [1, \beta] \\ y_j \in S_2}} (1 - p(v, y_j)) \right) \mid S_2 \subseteq U_2 \text{ is of size } c_2 \right\}, \\ Z_\gamma &= \max \left\{ \mathcal{C}(U_0, S_0) \mid S_0 \subseteq U_0 \text{ is of size } c_0, \text{ and } d = I_{z \in S_0} \right\}. \end{aligned}$$

So, the maximization considered in Eq. (6) is equivalent to the following optimization:

$$\max_{\substack{\gamma = (d, c_0, c_1, c_2) \in \Gamma \\ (a, b) \in \mathcal{L}_\gamma^1, (\bar{a}, \bar{b}) \in \mathcal{L}_\gamma^2}} \omega_v + Z_\gamma + \left(\omega_v (1 - d \cdot p(v, z)) \right) (a + \bar{a} - b\bar{b}). \quad (7)$$

In the next lemma we show that optimizing the above expression is equivalent to solving $|\Gamma| = O(k^2)$ different 2-SPM problems (each of size $O(n^{k/2})$).

► **Lemma 25.** Let $A = ((a_1, b_1), \dots, (a_N, b_N))$ and $\bar{A} = ((\bar{a}_1, \bar{b}_1), \dots, (\bar{a}_N, \bar{b}_N))$ be two arrays. Then, solving the maximization problem $\max_{i_0, j_0} (a_{i_0} + \bar{a}_{j_0} - b_{i_0} \bar{b}_{j_0})$, can be transformed in linear time to the following equivalent 2-SPM:

$$\mathcal{L} = ((Q + a_1, Rb_1), \dots, (Q + a_N, Rb_N), (-Q + \bar{a}_1, R^{-1}\bar{b}_1), \dots, (-Q + \bar{a}_N, R^{-1}\bar{b}_N)) ,$$

where $Q = \max_{i,j} (b_i \bar{b}_j) + 2 \max_{i,j} (a_i + \bar{a}_j)$ and $R = \sqrt{4Q / \min_i (b_i)^2}$.

Proof. Consider the following 2-SPMs obtained by two equal partitions of \mathcal{L} :

$$\begin{aligned} \mathcal{L}^1 &= ((Q + a_1, Rb_1), \dots, (Q + a_N, Rb_N)) , \text{ and} \\ \mathcal{L}^2 &= (-Q + \bar{a}_1, R^{-1}\bar{b}_1), \dots, (-Q + \bar{a}_N, R^{-1}\bar{b}_N) . \end{aligned}$$

Observe that the optimal value of \mathcal{L}^1 is bounded above by $2Q + 2 \max_i (a_i) - R^2 \min_i (\bar{b}_i)^2$ which is strictly less than $-Q$. Similarly, the optimal value of \mathcal{L}^2 is bounded above by $2Q + 2 \max_i (\bar{a}_i) - \min_i (\bar{b}_i)^2 / R^2$, which is again strictly less than $-Q$.

Now the answer to the optimization problem $\max_{i_0, j_0} (a_{i_0} + \bar{a}_{j_0} - b_{i_0} \bar{b}_{j_0})$ is at least $-Q$. This clearly shows that the solution to \mathcal{L} cannot be obtained by its restrictions \mathcal{L}^1 and \mathcal{L}^2 . Hence, the maximization problem $\max_{i_0, j_0} (a_{i_0} + \bar{a}_{j_0} - b_{i_0} \bar{b}_{j_0})$ is equivalent to solving the 2-SPM \mathcal{L} . ◀

Proof of Theorem 4. The time to compute $\mathcal{L}_\gamma^1, \mathcal{L}_\gamma^2, Z_\gamma$, for a given γ , is $\tilde{O}(n^{k/2})$. By transformation presented in Lemma 25, it follows that the total time required to optimize the expression in Eq. (7) is $k^{O(1)} \cdot n^{c \lceil k/2 \rceil}$, which is at most $O((\delta n)^{c \lceil k/2 \rceil + 1})$, for some constant $\delta \geq 1$. Now recall that Eq. (7) provides an optimal solution assuming (v, q) is a valid pair, and v is not contained in optimal S . Even when (v, q) is a valid pair, and v is contained in the optimal S , the time complexity turns out to be $O(k^2 \cdot n^{k/2})$. Iterating over all choices of pair (v, q) incurs an additional multiplicative factor of n in the runtime. ◀

6 Parameterization based on graph structure

In this section, we state our results on structural parameterizations of the Uni-PBDS problem. First, following the approach of Arnborg et al. [6], we formulate the MSOL formula for the Uni-PBDS problem where the quantifier rank of the formula is $O(k)$ (outlined in Appendix C.1). This indeed yields an FPT algorithm for the Uni-PBDS problem parameterized by budget k and the treewidth of the input graph.

In addition, we show that the Uni-PBDS problem is FPT for the budget parameter on apex-minor-free graphs. In particular we show that, for any integer k , and any n -vertex weighted apex-minor free graph with uniform edge probability, the Uni-PBDS problem can be solved in time $(2^{O(\sqrt{k} \log k)} n^{O(1)})$.

Due to space constraints, we defer our complete constructions and proofs to the upcoming full-version of the paper.

References

- 1 A. Abboud and K. Lewi. Exact weight subgraphs and the k-sum conjecture. In *Proc. 40th Int. Colloq. on Automata, Languages, and Programming (ICALP)*, pages 1–12. Springer, 2013.
- 2 A. Abboud, K. Lewi, and R. Williams. Losing weight by gaining edges. In *Proc. 22th European Symp. on Algorithms (ESA)*, pages 1–12. Springer, 2014.
- 3 J. Alber, M. R. Fellows, and R. Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.

- 4 J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. *J. Algorithms*, 52(1):26–56, 2004. doi:10.1016/j.jalgor.2004.03.005.
- 5 T.M. Apostol. *Calculus*. Number v. 1 in Blaisdell book in pure and applied mathematics. Blaisdell Pub. Co., 1969.
- 6 S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12:308–340, 1991.
- 7 S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth. Predicting protein complex membership using probabilistic network reliability. *Genome Research*, 14 6:1170–5, 2004.
- 8 J. Añez, T. De La Barra, and B. Pérez. Dual graph representation of transport networks. *Transportation Research Part B: Methodological*, 30(3):209–216, 1996. doi:10.1016/0191-2615(95)00024-0.
- 9 M. O. Ball. Complexity of network reliability computations. *Networks*, 10(2):153–165, 1980. doi:10.1002/net.3230100206.
- 10 M. O. Ball and J. S. Provan. Calculating bounds on reachability and connectedness in stochastic networks. *Networks*, 13(2):253–278, 1983. doi:10.1002/net.3230130210.
- 11 F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. Core decomposition of uncertain graphs. In *Proc. 20th ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 1316–1325, 2014.
- 12 C. J. Colbourn and G. Xue. A linear time algorithm for computing the most reliable source on a series-parallel graph with unreliable edges. *Theoretical Computer Science*, 209(1):331–345, 1998. doi:10.1016/S0304-3975(97)00124-2.
- 13 B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.
- 14 B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encycl. Mathematics and Its Applications*. Cambridge Univ. Press, 2012.
- 15 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 16 M. S. Daskin. A maximum expected covering location model: Formulation, properties and heuristic solution. *Transportation Science*, 17(1):48–70, 1983. URL: <http://EconPapers.repec.org/RePEc:inm:ortrsc:v:17:y:1983:i:1:p:48-70>.
- 17 Erik D. Demaine, Fedor V. Fomin, MohammadTaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H -minor-free graphs. *J. ACM*, 52(6):866–893, 2005.
- 18 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- 19 W. Ding. Computing the most reliable source on stochastic ring networks. In *2009 WRI World Congress on Software Engineering*, volume 1, pages 345–347, May 2009. doi:10.1109/WCSE.2009.31.
- 20 W. Ding. Extended most reliable source on an unreliable general network. In *2011 International Conference on Internet Computing and Information Services*, pages 529–533, September 2011. doi:10.1109/ICICIS.2011.138.
- 21 W. Ding and G. Xue. A linear time algorithm for computing a most reliable source on a tree network with faulty nodes. *Theoretical Computer Science*, 412(3):225–232, 2011. Combinatorial Optimization and Applications. doi:10.1016/j.tcs.2009.08.003.
- 22 Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, New York, NY, USA, 2001. ACM. doi:10.1145/502512.502525.

- 23 Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research, Dagstuhl Workshop, February 2-8, 1992*, pages 191–225, 1992.
- 24 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
- 25 P. G. Drange, M. S. Dregi, F. V. Fomin, S. Kreuzer, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, F. Reidl, F. S. Villaamil, S. Saurabh, S. Siebertz, and S. Sikdar. Kernelization and sparseness: the case of dominating set. In *Proc. 33rd Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 31:1–31:14, 2016.
- 26 T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihalák, and R. Raman. Computing minimum spanning trees with uncertainty. In *STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings*, pages 277–288, 2008. doi:10.4230/LIPIcs.STACS.2008.1358.
- 27 J. R. Evans. Maximum flow in probabilistic graphs—the discrete case. *Networks*, 6(2):161–183, 1976. doi:10.1002/net.3230060208.
- 28 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Subexponential algorithms for partial cover problems. *Inf. Process. Lett.*, 111(16):814–818, 2011. doi:10.1016/j.ipl.2011.05.016.
- 29 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 503–510. SIAM, 2010.
- 30 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Kernels for (connected) dominating set on graphs with excluded topological minors. *ACM Trans. Algorithms*, 14(1):6:1–6:31, 2018. doi:10.1145/3155298.
- 31 H. Frank and S. Hakimi. Probabilistic flows through a communication network. *IEEE Transactions on Circuit Theory*, 12(3):413–414, September 1965. doi:10.1109/TCT.1965.1082452.
- 32 M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable graphs. In *Proc. 26th Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 331–340, 1999.
- 33 Heng Guo and Mark Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3):964–978, 2019. doi:10.1137/18M1201846.
- 34 R. Hassin, R. Ravi, and F. S. Salman. *Tractable Cases of Facility Location on a Network with a Linear Reliability Order of Links*, pages 275–276. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- 35 R. Hassin, R. Ravi, and F. S. Salman. Multiple facility location on a network with linear reliability order of edges. *Journal of Combinatorial Optimization*, pages 1–25, 2017.
- 36 Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., Boston, MA, USA, 1997.
- 37 M. Hua and J. Pei. Probabilistic path queries in road networks: Traffic uncertainty aware path selection. In *Proc. 13th ACM Conf. on Extending Database Technology (EDBT)*, pages 347–358, 2010.
- 38 R. M. Karp and M. Luby. Monte-carlo algorithms for the planar multiterminal network reliability problem. *J. Complexity*, 1(1):45–64, 1985. doi:10.1016/0885-064X(85)90021-4.
- 39 D. Kempe, J. M. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. Ninth ACM Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.
- 40 Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, 1999. doi:10.1016/S0020-0190(99)00031-9.
- 41 T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

- 42 J. Kneis, D. Mölle, and P. Rossmanith. Partial vs. complete domination: T-dominating set. In *Proc. 33rd Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 367–376. Springer-Verlag, 2007.
- 43 Emanuel Melachrinoudis and Mary E. Helander. A single facility location problem on a tree with unreliable edges. *Networks*, 27(4):219–237, 1996. doi:10.1002/(SICI)1097-0037(199605)27:3.
- 44 N. S. Narayanaswamy, M. Nasre, and R. Vijayaragunathan. Facility location on planar graphs with unreliable links. In *Proc. 13th Computer Science Symp in Russia (CSR)*, pages 269–281, 2018.
- 45 N. S. Narayanaswamy and R. Vijayaragunathan. Parameterized optimization in uncertain graphs - A survey and some results. *Algorithms*, 13(1):3, 2020.
- 46 G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978. doi:10.1007/BF01588971.
- 47 M. Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proc. 42nd ACM Symp. on Theory of Computing (STOC)*, pages 603–610, 2010.
- 48 M. Patrascu and R. Williams. On the possibility of faster SAT algorithms. In *Proc. 21st ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1065–1075, 2010.
- 49 Y. Peng, Y. Zhang, W. Zhang, X. Lin, and L. Qin. Efficient probabilistic k-core computation on uncertain graphs. In *Proc. 34th IEEE Conf. on Data Engineering (ICDE)*, pages 1192–1203, 2018.
- 50 Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Transactions on Algorithms*, 9(1):11, 2012. doi:10.1145/2390176.2390187.
- 51 J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983. doi:10.1137/0212053.
- 52 G. Swamynathan, C. Wilson, B. Boe, K. C. Almeroth, and B. Y. Zhao. Do social networks improve e-commerce?: a study on social marketplaces. In *Proc. 1st Workshop on Online Social Networks (WOSN)*, pages 1–6, 2008.
- 53 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979. doi:10.1137/0208032.
- 54 Douglas R. White and Frank Harary. The cohesiveness of blocks in social networks: Node connectivity and conditional density. *Sociological Methodology*, 31(1):305–359, 2001. doi:10.1111/0081-1750.00098.
- 55 Zhaonian Zou and Jianzhong Li. Structural-context similarities for uncertain graphs. In *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, pages 1325–1330, 2013. doi:10.1109/ICDM.2013.22.

A Remainder of Section 4

A.1 Figures illustrated in the reduction

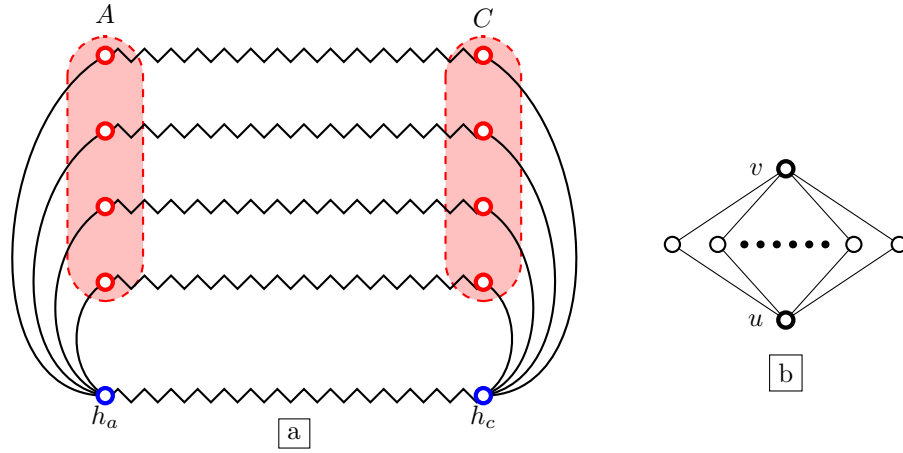


Figure 2 (a) The gadget \mathcal{I} for $n = 4$. (b) The gadget \mathcal{D} . The zigzag edges in \mathcal{I} between two vertices u and v is replaced by the gadget $\mathcal{D}_{u,v}$.

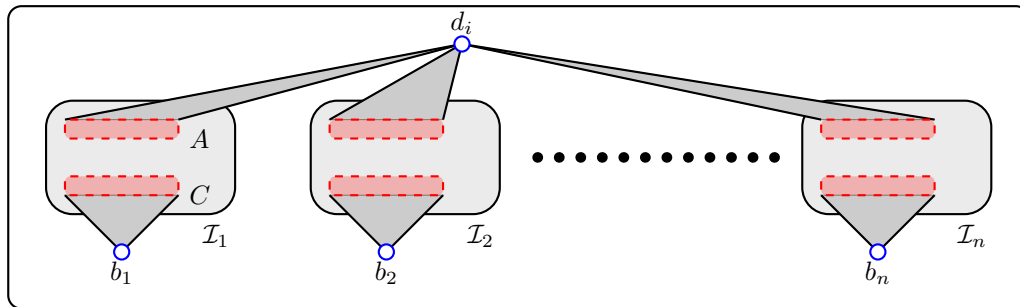


Figure 3 Illustration of a vertex block \mathcal{G}_i for a $V_i, i \in [k]$. Note the n \mathcal{I} gadgets for the n vertices in V_i . Similarly, an edge block $\mathcal{G}_{i,j}$ for some $1 \leq i < j \leq k$ has $|E_{i,j}|$ -many \mathcal{I} gadgets.

B Deferred Details from Section 5.4

B.1 Solving PBDS exactly on Trees

► **Reminder of Lemma 24.** For any set S of size k in \mathcal{T} , there exists a node $v \in \mathcal{T}$ and an index $q \in [1, \deg(v)]$ such that the cardinality of sets: $S \cap (\cup_{i \leq q} T_v^i)$, $S \cap (T_v^q)$, and $S \cap (\cup_{i \geq q} T_v^i)$, are all bounded by $k/2$.

Proof. We first show that there exists a node v in \mathcal{T} satisfying the property $|S \cap T_v^i| \leq k/2$, for each $i \in [1, \deg(v)]$. Consider a node $u \in \mathcal{T}$. If u satisfies the above mentioned property then we are done. Otherwise, there exists an index $j \in [1, \deg(u)]$ for which $|S \cap T_u^j| \geq k/2$. This implies the number of elements of S lying in $\{u\} \cup (T_u^1 \cup \dots \cup T_u^{j-1}) \cup (T_u^{j+1} \cup \dots \cup T_u^{\deg(u)})$ is at most $k/2$. In such a case we replace u by its j^{th} neighbor. Repeating the process eventually leads to the required node v .

32:22 Budgeted Dominating Sets in Uncertain Graphs

$$\text{Uni-PBDS} = \exists X \exists Y_0 \exists Y_1 \cdots \exists Y_k \left(\begin{aligned} & \text{PART}(X, Y_0, Y_1, \dots, Y_k) \wedge \\ & \forall x \forall y \left((y \in Y_0 \wedge x \in X) \rightarrow \neg(\text{adj}(x, y)) \right) \\ & \forall y \left(\bigwedge_{i=1}^k (y \in Y_i \rightarrow \text{INC}_i(y, X)) \right) \end{aligned} \right)$$

► **Lemma 26.** *The quantifier rank of Uni-PBDS is $O(k)$.*

Proof. There are $k + 2$ initial quantifiers for the sets X, Y_0, Y_1, \dots, Y_k . For two MSOL formulas ϕ and ψ with quantifier rank $\text{qr}(\phi)$ and $\text{qr}(\psi)$, respectively, $\text{qr}(\phi \wedge \psi) = \text{qr}(\phi \vee \psi) = \max\{\text{qr}(\phi), \text{qr}(\psi)\}$. Therefore, $\text{qr}(\text{Uni-PBDS})$ is bounded as follows:

$$\begin{aligned} \text{qr}(\text{Uni-PBDS}) &= k + 2 + \max\{\text{qr}(\text{PART}), 1 + \text{qr}(\text{INC})\} \\ &= k + 2 + \max\{1, 1 + \max\{\text{qr}(\text{SIZE}), 2\}\} \\ &\leq k + 2 + k = 2k + 3 = O(k) \end{aligned} \quad \blacktriangleleft$$

We now show that the Uni-PBDS problem is fixed-parameter tractable in parameters k and treewidth by expressing the maximization problem on the MSOL formula as a minor variation of *extended monadic second-order extremum problem* as described by Arnborg et. al. [6].

Proof of Theorem 8. For each $0 \leq i \leq k$, define the weight function w^i associated with the set variable Y_i as follows: for each $v \in V(G)$, $w_v^i = (1 - (1 - p)^i)w(v)$. The difference between the weight function in [6] and our problem is that in their paper $w(v)$ is considered to be constant value, for all vertices, for the set variable Y_i . Observe, however, that the running time of their algorithm does not change as long as w_v^i can be computed in polynomial time, which is the case in our definition. Therefore, our maximization problem is now formulated as a variant of the EMS maximization problem in [6]:

$$\text{Maximize } \sum_{u \in X} w(u) + \sum_{i=0}^k \sum_{u \in Y_i} w_v^i \cdot y_v^i \text{ over partitions } (X, Y_0, Y_1, \dots, Y_k) \text{ satisfying Uni-PBDS}$$

Using Theorem 5.6 in [6] along with the additional observation, we make, that w_v^i can be efficiently computed, an optimal solution for the Uni-PBDS problem can be computed in time $f(\text{qr}(\text{Uni-PBDS}), w) \cdot \text{poly}(n)$, where $f(\text{qr}(\text{Uni-PBDS}), w)$ is a function which does not depend on n - it depends only on the quantifier rank of Uni-PBDS and the treewidth. By Lemma 26, $\text{qr}(\text{Uni-PBDS}) = O(k)$, and thus by [6], $f(\text{qr}(\text{Uni-PBDS}), w) = f(O(k), w)$. This shows that Uni-PBDS is FPT with respect the parameters k and treewidth. Hence the theorem is proved. \blacktriangleleft

On the Complexity of the Escape Problem for Linear Dynamical Systems over Compact Semialgebraic Sets

Julian D’Costa ✉

Department of Computer Science, University of Oxford, UK

Engel Lefaucheu ✉ 

Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany

Eike Neumann ✉

Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany

Joël Ouaknine ✉ 

Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany

James Worrell ✉

Department of Computer Science, University of Oxford, UK

Abstract

We study the computational complexity of the Escape Problem for discrete-time linear dynamical systems over compact semialgebraic sets, or equivalently the Termination Problem for affine loops with compact semialgebraic guard sets. Consider the fragment of the theory of the reals consisting of negation-free $\exists\forall$ -sentences without strict inequalities. We derive several equivalent characterisations of the associated complexity class which demonstrate its robustness and illustrate its expressive power. We show that the Compact Escape Problem is complete for this class.

2012 ACM Subject Classification Theory of computation \rightarrow Logic and verification

Keywords and phrases Discrete linear dynamical systems, Program termination, Compact semialgebraic sets, Theory of the reals

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.33

Related Version *Full Version:* <https://arxiv.org/abs/2107.02060> [13]

Funding *Julian D’Costa:* emmy.network foundation under the aegis of the Fondation de Luxembourg. *Joël Ouaknine:* ERC grant AVS-ISS (648701), and DFG grant 389792660 as part of TRR 248 (see <https://perspicuous-computing.science>).

Joël Ouaknine is also affiliated with Keble College, Oxford as **emmy.network** Fellow.

James Worrell: EPSRC Fellowship EP/N008197/1.

1 Introduction

In ambient space \mathbb{R}^n , a *discrete linear dynamical system* is an orbit $(X_t)_{t \in \mathbb{N}}$ defined by an initial vector X_0 and a matrix A through the recursion $X_{t+1} = AX_t$. Linear dynamical systems are fundamental models across science and engineering, and the computability and complexity of decision problems concerning them are of both theoretical and practical importance.

In the study of dynamical systems, especially in control theory, considerable attention has been given to analysing *invariant sets*, i.e., subsets of \mathbb{R}^n from which no trajectory can escape; see, e.g., [10, 5, 2, 21]. Our focus in the present paper is on sets with the dual property that *no trajectory remains trapped*. Such sets play a key role in analysing *liveness* properties:



© Julian D’Costa, Engel Lefaucheu, Eike Neumann, Joël Ouaknine, and James Worrell; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 33; pp. 33:1–33:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

progress is ensured by guaranteeing that all trajectories (i.e., from any initial starting point) must eventually reach a point at which they “escape” (temporarily or permanently) the set in question, thereby forcing a system transition to take place.

More precisely, given a rational matrix A and a semialgebraic set $K \subseteq \mathbb{R}^n$, one may consider the *Discrete Escape Problem (DEP)* which asks, for all starting points X_0 in K , whether the corresponding orbit of the discrete linear dynamical system $(X_t)_{t \in \mathbb{N}}$ eventually escapes K . By “escaping” K , we simply mean going outside of K – we are unconcerned whether the trajectory might re-enter K at a later time.

The restriction of DEP to the case in which K is a *convex polytope* – alternately known as *termination of linear programs* over either the reals or the rationals – was already studied and shown decidable in the seminal papers [24, 6], albeit with no complexity bounds nor upper bounds on the number of iterations required to escape.

In this paper we study the *Compact Escape Problem (CEP)*, a version of DEP where in addition we assume that the semialgebraic set K is compact. In practice, of course, this is usually not a burdensome restriction; in most cyber-physical systems applications, for instance, all relevant sets will be compact (see, e.g., [1]).

CEP was recently shown to be decidable for arbitrary compact semialgebraic sets in [18], via *non-constructive* methods; consequently – as pointed out in that paper – no non-trivial complexity bounds could be given. The main contribution of the present work is to precisely pin down the complexity of CEP in terms of the first-order theory of the reals; more precisely, we identify a natural fragment for which CEP is complete.

Recall that the theory of the reals is concerned with the structure \mathbb{R} over the signature $\langle \mathbb{Z}, +, \times, \leq, < \rangle$. Tarski famously showed that this theory is decidable and admits quantifier elimination, with state-of-the-art techniques based on Collins’s *Cylindrical Algebraic Decomposition* [12] that have complexity doubly exponential in the number of quantifiers. Asymptotically faster but arguably impractical quantifier elimination algorithms due to [14, 16, 20] have running time doubly exponential in the number of quantifier alternations, singly exponential in the dimension, and polynomial in the rest of the data. The *existential* fragment of the theory of the reals was famously shown to lie between NP and PSPACE in [9].

In this paper, we consider the class of formulas consisting of positive Boolean combinations of non-strict polynomial inequalities prefixed by a single alternation of a block of existential and a block of universal quantifiers. Let us denote by $\exists \forall_{\leq} \mathbb{R}$ the complexity class of all problems reducible in polynomial time to the decision problem for this fragment. Using sophisticated results from real algebraic geometry we show that $\exists \forall_{\leq} \mathbb{R}$ corresponds to the decision problem for another fragment of $\exists \forall$ -sentences in which the quantifiers are restricted to range over compact sets, a result of independent interest. Finally, using techniques from Diophantine approximation and algebraic number theory we show that the Compact Escape Problem is complete for this class.

1.1 Overview

We formally define the Compact Escape Problem (CEP) as the following decision problem:

Given as input

- A matrix $A \in \mathbb{Q}^{n \times n}$ with rational entries,
 - A list \mathcal{P} of polynomials in $\mathbb{Z}[x_1, \dots, x_n]$,
 - A propositional formula $\Phi(x_1, \dots, x_n)$ which combines atomic predicates of the form $P(x_1, \dots, x_n) \leq 0$ with $P \in \mathcal{P}$ by means of the propositional connectives \vee and \wedge ,
- subject to the promise that the set $K = \{x \in \mathbb{R}^n \mid \Phi(x)\}$ is compact, decide whether for all $x \in K$ there exists $k \in \mathbb{N}$ such that $A^k x \notin K$.*

We assume that the polynomials P_j in the list \mathcal{P} are encoded as lists $\langle (\alpha_{j,k}, c_{j,k}) \rangle_{k=1, \dots, s_j}$ of pairs of multi-indexes $\alpha_{j,k} \in \mathbb{N}^n$, whose entries are encoded in unary, and coefficients $c_{j,k} \in \mathbb{Z}$, encoded in binary, such that

$$P_j(x_1, \dots, x_n) = \sum_{k=1}^{s_j} c_{j,k}(x_1, \dots, x_n)^{\alpha_{j,k}}. \quad (1)$$

Note that the analogous problem for affine maps $x \mapsto Ax + b$ reduces to CEP, as a point $x \in K$ escapes the compact set K under iterations of the affine map $Ax + b$ if and only if the point $(x, 1) \in K \times \{1\}$ escapes $K \times \{1\}$ under iterations of the linear map $B(x, z) = Ax + bz$.

We capture the computational complexity of this decision problem by showing that it is equivalent to the decision problem for a fragment of the theory of the reals.

Let $\exists\forall_{\leq}\mathbb{R}$ denote the decision problem for sentences of the form

$$\exists X \in \mathbb{R}^n. \forall Y \in \mathbb{R}^m. (\Phi_{0,\leq}(X, Y)), \quad (2)$$

where $\Phi_{0,\leq}$ is a positive Boolean combination of non-strict polynomial inequalities. Evidently, this class lies between the existential fragment of the theory of the reals (without restriction on the types of inequalities) and the full $\exists\forall$ -fragment.

The main result of this paper is the following:

► **Theorem 1.** *The compact escape problem is complete for the complexity class $\exists\forall_{\leq}\mathbb{R}$.*

The proof consists of three steps:

First, we show that for any sentence of the form

$$\exists X \in [-1, 1]^n. \forall Y \in [-1, 1]^m. (\Phi_{0,\leq}(X, Y)), \quad (3)$$

where $\Phi_{0,\leq}$ is a positive Boolean combination of non-strict polynomial inequalities, one can compute a matrix $A \in \mathbb{Q}^{(n+2m) \times (n+2m)}$ and a compact set $K \subseteq \mathbb{R}^{n+2m}$ such that (A, K) is a negative instance of the compact escape problem if and only if (3) holds true.

Secondly, given any instance (A, K) with $A \in \mathbb{Q}^{n \times n}$ and $K \subseteq \mathbb{R}^n$ we can compute in polynomial time a sentence of the form

$$\exists X \in [-1, 1]^m. \forall Y \in [-1, 1]^\ell. (\Psi_{0,\leq}(Y) \rightarrow \Phi_{0,\leq}(X, Y)), \quad (4)$$

where $\Psi_{0,\leq}$ and $\Phi_{0,\leq}$ are a positive Boolean combination of non-strict polynomial inequalities, such that (4) holds true if and only if (A, K) is a negative instance of the compact escape problem.

Finally, we prove that the decision problems for sentences of the form (2), (3), and (4) are all equivalent.

2 Preliminaries

2.1 Fragments of the theory of the reals

The statement and proof of Theorem 1 require complexity classes induced by decision problems for fragments of the the first-order theory of the reals. The main goal of this subsection is to formally define these complexity classes.

Thus, let \mathcal{L} be the first-order language with signature $\langle \mathbb{Z}, +, \times, <, \leq \rangle$, propositional connectives, \wedge and \vee , and quantifiers \exists and \forall . For complexity purposes, we assume that integer constants are encoded in binary. See, e.g., [23, 25] for an introduction to first-order

33:4 On the Complexity of the Compact Escape Problem

logic. We interpret all formulas in \mathcal{L} in the structure of real numbers. Thus, we say that two formulas are equivalent if their interpretations in \mathbb{R} are equivalent. The restriction to the connectives \vee and \wedge is of course insubstantial, and we will make free use of the connectives \neg and \rightarrow throughout this paper, understanding them as syntactic sugar.

Let QFF denote the set of quantifier-free formulas in \mathcal{L} . Let QFF_{\leq} (resp. $\text{QFF}_{<}$) denote the subset of QFF consisting of those formulas that do not contain the relational symbol “ $<$ ” (resp. “ \leq ”). Note that the negation of a QFF_{\leq} -formula is a $\text{QFF}_{<}$ -formula and vice versa.

We define the sets of formulas $\Sigma_{n,\leq}$ and $\Pi_{n,\leq}$ inductively as follows:

1. Let $\Sigma_{0,\leq} = \Pi_{0,\leq} = \text{QFF}_{\leq}$.
2. A formula $\Psi(y_1, \dots, y_s)$ belongs to $\Sigma_{n+1,\leq}$ if and only if it is of the form

$$\Psi(y_1, \dots, y_s) = (\exists x_1) \dots (\exists x_t) \cdot \Phi(x_1, \dots, x_t, y_1, \dots, y_s),$$

where Φ belongs to $\Pi_{n,\leq}$.

3. Dually, a formula $\Psi(y_1, \dots, y_s)$ belongs to $\Pi_{n+1,\leq}$ if and only if it is of the form

$$\Psi(y_1, \dots, y_s) = (\forall x_1) \dots (\forall x_t) \cdot \Phi(x_1, \dots, x_t, y_1, \dots, y_s),$$

where Φ belongs to $\Sigma_{n,\leq}$.

We define $\Sigma_{n,<}$ and $\Pi_{n,<}$ (resp. Σ_n and Π_n) analogously, starting with $\text{QFF}_{<}$ -formulas (resp. QFF -formulas).

By convention we denote vectors of variables $X = (x_1, \dots, x_t)$ by upper case letters and introduce the shorthand notations $\exists X$ and $\forall X$ for blocks of quantifiers $(\exists x_1) \dots (\exists x_t)$ and $(\forall x_1) \dots (\forall x_t)$. Recall that a first-order formula Φ is called a sentence if it does not contain any free variables.

The *decision problem* for a class \mathcal{C} of first-order formulas in the language \mathcal{L} is the following: Given a sentence that belongs to \mathcal{C} decide whether the sentence holds true in the universe of real numbers.

It is natural to ask how the decision problems for the classes we have introduced above are related with respect to polynomial-time reductions. By taking the negation of formulas it is easy to see that the decision problem for Σ_n is equivalent to that of Π_n , the decision problem for $\Sigma_{n,\leq}$ is equivalent to that of $\Pi_{n,<}$, and the decision problem for $\Sigma_{n,<}$ is equivalent to that of $\Pi_{n,\leq}$. As such it suffices to consider the “ Σ ”-classes in the following.

By a standard trick, any QFF-formula $\Phi(X)$ with free variables X can be converted in polynomial time into an equivalent formula $\exists Y. f(X, Y) = 0$ where f is a single polynomial. It follows that if n is odd then the decision problems for the classes Σ_n and $\Sigma_{n,\leq}$ are polynomial-time equivalent and if n is even then the decision problems for the classes Σ_n and $\Sigma_{n,<}$ are polynomial-time equivalent.

Of course, for $n = 0$ the decision problem is trivial for all three classes. For $n = 1$ we have the following remarkable result:

► **Theorem 2** ([22]). *The decision problems for Σ_1 and $\Sigma_{1,<}$ are polynomial-time equivalent.*

We thus have polynomial-time reductions for decision problems as indicated below:

$$(\Sigma_0 \equiv \Sigma_{0,\leq} \equiv \Sigma_{0,<}) \rightarrow (\Sigma_1 \equiv \Sigma_{1,\leq} \equiv \Sigma_{1,<}) \rightarrow \Sigma_{2,\leq} \rightarrow (\Sigma_2 \equiv \Sigma_{2,<}) \rightarrow \Sigma_{3,<} \rightarrow \dots$$

It is open to the best of our knowledge whether there exists a reduction of the decision problem for Σ_2 to that of $\Sigma_{2,\leq}$. The techniques from [22] do not seem to carry over to higher orders of quantifier alternations.

We study the decision problem for the class $\Sigma_{2,\leq}$ in greater detail. Let us denote by $\exists\forall_{\leq}\mathbb{R}$ the complexity class of all problems reducible in polynomial time to this decision problem. To demonstrate the robustness of this complexity class and gauge its computational

power we give a number of equivalent characterisations. It turns out that, somewhat surprisingly, the decision problem for $\Sigma_{2,\leq}$ -sentences is equivalent to the decision problem for exists-forall-sentences whose quantifiers are restricted to range over compact sets.

Let $X = (x_1, \dots, x_n)$ be a vector of variables. Let y be a variable or a constant. We write $|X| \leq y$ as an abbreviation for the formula $\bigwedge_{j=1}^n (-y \leq x_j \leq y)$. Of course, this syntactic construct will only have the intended semantics if our context ensures that $y \geq 0$, and we will only use it in such situations.

Write $I = [-1, 1]$. Let $\Phi_0(X, Y, Z)$ be a quantifier-free formula in \mathcal{L} . We introduce the syntactic abbreviation

$$\exists X \in I^n. \forall Y \in I^m. (\Phi_0(X, Y, Z))$$

for the formula

$$\exists X \in \mathbb{R}^n. \forall Y \in \mathbb{R}^m. (|Y| > 1 \vee (|X| \leq 1 \wedge \Phi_0(X, Y, Z)))$$

in the language \mathcal{L} .

We have the following result, whose proof is the focus of Section 3:

► **Theorem 3.** *The decision problems for the following three classes of sentences are equivalent with respect to polynomial-time reduction:*

1. *The class $\Sigma_{2,\leq}$, consisting of sentences of the form*

$$\exists X \in \mathbb{R}^m. \forall Y \in \mathbb{R}^n. (\Phi_{0,\leq}(X, Y)),$$

where $\Phi_{0,\leq}$ is a QFF_{\leq} -formula.

2. *The class $\mathfrak{b}\text{-}\Sigma_{2,\leq}$, consisting of sentences of the form*

$$\exists X \in I^m. \forall Y \in I^n. (\Phi_{0,\leq}(X, Y)),$$

where $\Phi_{0,\leq}$ is a QFF_{\leq} -formula.

3. *The class $\mathfrak{b}\text{-}\Sigma_{2,\leq}^{++}$, consisting of sentences of the form*

$$\exists X \in I^m. \forall Y \in I^n. (\Psi_{0,\leq}(Y) \rightarrow \Phi_{0,\leq}(X, Y)),$$

where $\Phi_{0,\leq}$ and $\Psi_{0,\leq}$ are QFF_{\leq} -formulas.

It is obvious that the decision problem for $\mathfrak{b}\text{-}\Sigma_{2,\leq}$ -sentences reduces to that of $\mathfrak{b}\text{-}\Sigma_{2,\leq}^{++}$ -sentences. Note however that it is not clear that a reduction should exist in either direction between $\Sigma_{2,\leq}$ and $\mathfrak{b}\text{-}\Sigma_{2,\leq}$. On the one hand, the latter class only allows for quantification over bounded sets, which seems to make it more restrictive. On the other hand, $\mathfrak{b}\text{-}\Sigma_{2,\leq}$ -sentences involve strict inequalities and hence do not belong to the class $\Sigma_{2,\leq}$. Let us denote by $\mathfrak{b}\text{-}\exists\forall_{\leq}\mathbb{R}$ and by $\mathfrak{b}\text{-}\exists\forall_{\leq}^{++}\mathbb{R}$ the complexity classes induced respectively by the decision problem for $\mathfrak{b}\text{-}\Sigma_{2,\leq}$ -sentences and by the decision problem for $\mathfrak{b}\text{-}\Sigma_{2,\leq}^{++}$ -sentences.

A remark is in order on the robustness of our definition of the class $\exists\forall_{\leq}\mathbb{R}$ under different encodings of polynomials. In practice it is common to encode a polynomial P as a list $\langle\langle\alpha_j, c_j\rangle\rangle_{j=1,\dots,m}$ where $\alpha_j \in \mathbb{N}^n$ are multi-indexes and $c_j \in \mathbb{Z}$ are integers satisfying (1). This is the encoding we have chosen in the definition of CEP. By contrast, the polynomials that occur in atomic predicates of a formula in the language \mathcal{L} are encoded as terms over the signature $\langle\mathbb{Z}, +, \times\rangle$. While one can translate the encoding (1) to a term over the signature $\langle\mathbb{Z}, +, \times\rangle$ in polynomial time, a term of size N can encode a polynomial whose number of non-zero coefficients grows exponentially in N , so that a polynomial-time translation in the

other direction is not possible in general. One may hence raise the justified objection that the reduction of CEP to the decision problem for $\Sigma_{2,\leq}$ sentences could hide an exponential overhead in the encoding of the polynomials. Moreover, in order to show $\exists\forall_{\leq}\mathbb{R}$ -hardness of CEP we need to convert a compact set which is encoded as a QFF $_{\leq}$ -formula into an equivalent formula whose atoms use the encoding (1). We show in Theorem 20 that we can efficiently convert any $\Sigma_{2,\leq}$ -sentence into an equivalent one whose atoms have degree at most 4. This resolves the issue, for a uniform bound on the degrees allows one to translate back and forth in polynomial time between the two encodings of polynomials. While an analogous result for Σ_2 -sentences (and, e.g., QFF $_{\leq}$ -formulas) is straightforward (see e.g. [22, Lemma 3.2] or the proof of Theorem 20 below for a proof idea), the argument becomes much more involved for $\Sigma_{2,\leq}$ -sentences. It relies on many of the results that are established in the sequel. Thus, for the majority of this paper we have to insist on our specific choice of encoding.

2.2 Mathematical tools

Our characterisation of the complexity class $\exists\forall_{\leq}\mathbb{R}$ requires two sophisticated results from effective real algebraic geometry: Singly exponential quantifier elimination and a doubly exponential bound on a ball meeting all components of a semialgebraic set. We use the following singly exponential quantifier elimination result given in [3]. For a historical overview on this type of result see [3, Chapter 14, Bibliographical Notes].

► **Theorem 4** ([3, Theorem 14.16]). *Let \mathcal{P} be a set of at most s polynomials with integer coefficients, each of degree at most d , in $k + n_1 + \dots + n_\ell$ variables. Let τ be a bound on the bitsize of the coefficients of all $P \in \mathcal{P}$. Let*

$$\Phi_\ell(Y) = (Q_1 X_1) \dots (Q_\ell X_\ell) \cdot (\Psi_0(Y, X_1, \dots, X_\ell)),$$

where $Q_j \in \{\exists, \forall\}$ are alternating blocks of quantifiers, be a formula over the language \mathcal{L} , all of whose atoms involve polynomials contained in \mathcal{P} . Assume that the size of the block of variables Y is k and that the size of the block of variables X_j is n_j .

Then there exists an equivalent quantifier-free formula

$$\omega_0(Y) = \bigvee_{i=1}^I \bigwedge_{j=1}^{J_i} \bigvee_{m=1}^{M_{i,j}} P_{i,j,m}(Y) \bowtie_{i,j,m} 0.$$

over \mathcal{L} , where:

1. $I \leq s^{(n_1+1)\dots(n_\ell+1)(k+1)} d^{O(n_1\dots n_\ell k)}$.
2. $J_i \leq s^{(n_1+1)\dots(n_\ell+1)} d^{O(n_1\dots n_\ell)}$.
3. $M_{i,j} \leq d^{O(n_1\dots n_\ell)}$.
4. The degrees of the polynomials $P_{i,j,m}$ are bounded by $d^{O(n_1\dots n_\ell)}$.
5. The bitsize of the coefficients of the polynomials $P_{i,j,m}$ is bounded by $\tau d^{O(n_1\dots n_\ell k)}$.

Recall that a *sign condition* on a family \mathcal{P} of polynomials in n variables is a mapping $\sigma: \mathcal{P} \rightarrow \{-1, 0, 1\}$. The *realisation* of a sign condition σ in \mathbb{R}^n is the set

$$\text{Real}(\sigma) = \{X \in \mathbb{R}^n \mid \forall P \in \mathcal{P}. \text{sign}(P(X)) = \sigma(P)\}.$$

A sign condition σ is called *realisable* if its realisation is non-empty. Equivalently, a sign condition is a formula over the language \mathcal{L} involving only conjunctions.

The next theorem is due to Vorobjov [26]. See also [15, Lemma 9] and [4, Theorem 4].

► **Theorem 5.** *There exists an integer constant β' with the following property: Let \mathcal{P} be a set of s polynomials with integer coefficients in n variables of degree at most $d \geq 2$. Assume that the bit-size of the coefficients of each polynomial in \mathcal{P} is at most τ . Then there exists a ball centred at the origin of radius at most*

$$2^{\tau d^{\beta'(n+1)}}$$

which intersects every connected component of every realisable sign condition on \mathcal{P} in \mathbb{R}^n .

Our proof of $\exists\forall\leq\mathbb{R}$ -completeness of CEP combines spectral methods with two well-known but nontrivial results on algebraic numbers. We require a version of Kronecker's theorem on simultaneous Diophantine approximation. See [19, Corollary 3.1] for a proof.

► **Theorem 6.** *Let $(\lambda_1, \dots, \lambda_m)$ be complex algebraic numbers of modulus 1. Consider the free Abelian group*

$$L = \{(n_1, \dots, n_m) \in \mathbb{Z}^m \mid \lambda_1^{n_1} \cdots \lambda_m^{n_m} = 1\}.$$

Let $(\beta_1, \dots, \beta_s)$ be a basis of L . Let $\mathbb{T}^m = \{(z_1, \dots, z_m) \in \mathbb{C}^m \mid |z_j| = 1\}$ denote the complex unit m -torus. Then the closure of the set $\{(\lambda_1^k, \dots, \lambda_m^k) \in \mathbb{T}^m \mid k \in \mathbb{N}\}$ is the set $S = \{(z_1, \dots, z_m) \in \mathbb{T}^m \mid \forall j \leq s. (z_1, \dots, z_m)^{\beta_j} = 1\}$.

Moreover, for all $\varepsilon > 0$ and all $(z_1, \dots, z_m) \in S$ there exist infinitely many indexes k such that $|\lambda_j^k - z_j| < \varepsilon$ for $j = 1, \dots, m$.

Moreover, the integer multiplicative relations between given complex algebraic numbers in the unit circle can be elicited in polynomial time. For a proof see [8, 17]. We assume the standard encoding of algebraic numbers, see [11] for details.

► **Theorem 7.** *Let $(\lambda_1, \dots, \lambda_m)$ be complex algebraic numbers of modulus 1. Consider the free Abelian group*

$$L = \{(n_1, \dots, n_m) \in \mathbb{Z}^m \mid \lambda_1^{n_1} \cdots \lambda_m^{n_m} = 1\}.$$

Then one can compute in polynomial time a basis $(\beta_1, \dots, \beta_s) \in (\mathbb{Z}^m)^s$ for L . Moreover, the integer entries of the basis elements β_j are bounded polynomially in the size of the encodings of $\lambda_1, \dots, \lambda_m$.

3 Proof of Theorem 3

Our proof of Theorem 3 will use Theorems 4 and 5. The latter are formulated in terms of the algebraic complexity of a family of polynomials. We will reformulate them in terms of the bitsize of a formula in the language \mathcal{L} .

The *matrix size* μ of a first-order formula

$$\Psi(Y) = (Q_1 X_1) \cdots (Q_\ell X_\ell). (\Phi_0(Y, X_1, \dots, X_\ell)),$$

where $Q_j \in \{\exists, \forall\}$ is the number of bits required to write down the quantifier-free part $\Phi_0(Y, X_1, \dots, X_\ell)$. The *dimensions* of the formula $\Psi(Y)$ are the numbers m, n_1, \dots, n_ℓ , where m is the dimension of Y . The *size* σ of the formula $\Psi(Y)$ is the number of bits required to write down the whole formula. Note that we have $\sigma = O(m + n_1 + \dots + n_\ell + \mu)$.

Observe that if $\Phi(X)$ is a QFF-formula of (matrix) size μ and $P(X) \bowtie 0$ is an atom of Φ then P has degree at most μ and its coefficients are bounded in bitsize by μ . The following is an immediate corollary to Theorem 4:

► **Theorem 8.** *There exists a constant α with the following property:*

Let

$$(Q_1 X_1) \dots (Q_\ell X_\ell) \cdot \Phi_0(Y, X_1, \dots, X_\ell)$$

be a first-order formula in the language \mathcal{L} of matrix size μ and with dimensions m, n_1, \dots, n_ℓ . Then there exists an equivalent quantifier-free formula $\Psi_0(Y)$ of size at most

$$\mu^{\alpha^{\ell+1} \cdot ((m+1) \cdot (n_1+1) \cdots (n_\ell+1))}.$$

Theorem 5 entails the following:

► **Corollary 9.** *There exists a constant β with the following property: Let $\Phi_0(X)$ be a quantifier-free formula in the language \mathcal{L} of matrix size μ and dimension $n \geq 1$. Then the sentence $\exists X \in \mathbb{R}^n. (\Phi_0(X))$ is equivalent to the sentence*

$$\exists X. \left(|X| \leq 2^{\mu^{\beta(n+1)}} \wedge \Phi_0(X) \right)$$

Theorem 8 and Corollary 9 will allow us to efficiently convert certain formulas into equivalent ones whose quantifiers range over bounded intervals of doubly exponential size in the input data. By the standard repeated squaring trick such formulas can further be efficiently converted into equivalent ones whose quantifiers range over the interval $I = [-1, 1]$. See Lemma 25 in Appendix A for a precise statement and proof.

3.1 Showing $\exists \forall_{\leq} \mathbb{R} \subseteq \mathbf{b}\text{-}\exists \forall_{\leq} \mathbb{R}$

We now show that the decision problem $\exists \forall_{\leq} \mathbb{R}$ reduces to $\mathbf{b}\text{-}\exists \forall_{\leq} \mathbb{R}$ in polynomial time.

We first bound the existential quantifier. This bound does not yet require the quantifier-free part of the sentence to involve only non-strict inequalities.

► **Lemma 10.** *Let $\exists X \in \mathbb{R}^n. \forall Y \in \mathbb{R}^m. (\Phi_0(X, Y))$. be a sentence over the language \mathcal{L} of matrix size μ . Then, denoting $I = [-1, 1]$, we can compute in polynomial time an equivalent sentence of the form*

$$\exists X \in I^{m+N}. \forall Y \in \mathbb{R}^m. (\Psi_0(X, Y)).$$

Proof. Consider the formula $\chi_1(X) = \forall Y \in \mathbb{R}^m. (\Phi_0(X, Y))$. By Theorem 8 this formula is equivalent to a quantifier-free formula $\chi_0(X)$ of size at most $\mu^{\alpha^2(n+1)(m+1)}$. By Corollary 9 the sentence $\exists X \in \mathbb{R}^n. (\chi_0(X))$ is equivalent to the sentence

$$\exists X \in \mathbb{R}^n. \left(|X| \leq 2^{\mu^{\alpha^2 \beta (n+1)^2 (m+1)}} \wedge \chi_0(X) \right).$$

Hence, our original sentence is equivalent to the sentence

$$\exists X \in \mathbb{R}^n. \forall Y \in \mathbb{R}^m. \left(|X| \leq 2^{\mu^{\alpha^2 \beta (n+1)^2 (m+1)}} \wedge \Phi_0(X, Y) \right).$$

Now, we can compute in polynomial time a positive integer N in unary such that we have $\mu^{\alpha^2 \beta (n+1)^2 (m+1)} \leq 2^N$. By (the proof of) Lemma 25 in Appendix A we obtain an equivalent sentence as claimed. ◀

Next we derive a similar bound for the universal quantifier in terms of the bound for the existential one. This will require the assumption that all inequalities are non-strict. The reason for this is the following simple continuity property of QFF $_{<}$ -formulas, which can fail for general formulas in the language \mathcal{L} :

► **Proposition 11.** *Let $\Phi_0(X)$ be a $\text{QFF}_{<}$ -formula with a vector of n free variables X . Assume that $x \in \mathbb{R}^n$ is such that $\Phi_0(x)$ holds true. Then there exists $\varepsilon > 0$ such that $\Phi_0(\tilde{x})$ holds true for all $\tilde{x} \in \mathbb{R}^n$ with $|x - \tilde{x}| < \varepsilon$.*

Proof. By structural induction on the formula Φ . The base case follows from the fact that polynomials are continuous functions. The induction steps are easy. ◀

► **Lemma 12.** *Let $B \in \mathbb{N}$ be a positive integer constant. Let*

$$\Psi = \forall X \in \mathbb{R}^n. \exists Y \in \mathbb{R}^m. (|X| > B \vee \Phi_0(X, Y))$$

be a $\Pi_{2,<}$ -sentence. Then the sentence Ψ holds true over the reals if and only if the sentence

$$\Psi' = \exists C \in \mathbb{R}. \forall X \in \mathbb{R}^n. \exists Y \in \mathbb{R}^m. (|X| > B \vee (Y \leq C \wedge \Phi(X, Y)))$$

holds true over the reals.

Proof. Clearly, Ψ' implies Ψ , so that if Ψ is false then Ψ' is false.

Suppose now that Ψ is true. Let $K = \{X \in \mathbb{R}^n \mid |X| \leq B\}$. Then, by assumption, for all $X \in K$ there exists $Y(X) \in \mathbb{R}^m$ such that $\Phi(X, Y(X))$ holds true. It follows from Proposition 11 that there exists $\varepsilon(X) > 0$ such that $\Phi(X', Y(X))$ holds true for all X' with $|X - X'| < \varepsilon(X)$. The set $\{\text{Ball}(X, \varepsilon(X)) \mid X \in K\}$, where $\text{Ball}(X, c)$ denotes the ball of radius c centered at X , is an open cover of K . The set K is compact, so that this cover has a finite subcover $\text{Ball}(X_1, \varepsilon(X_1)), \dots, \text{Ball}(X_s, \varepsilon(X_s))$. It follows that for all $X \in K$ there exists $j \in \{1, \dots, s\}$ such that $\Phi(X, Y(X_j))$ holds true. Thus, the formula Ψ' holds true with $C = \max\{|Y(X_1)|, \dots, |Y(X_s)|\}$. ◀

Note that the conclusion of Lemma 12 does not hold true in general for $\Pi_{2,\leq}$ -formulas. For instance, the formula

$$\forall x \in [-1, 1]. \exists y \in \mathbb{R}. (x^2(1 - xy) \leq 0)$$

is clearly true, but the formula

$$\exists C \in \mathbb{R}. \forall x \in [-1, 1]. \exists y \in [-C, C]. (x^2(1 - xy) \leq 0)$$

is clearly false.

► **Lemma 13.** *Given a sentence of the form*

$$\exists X \in I^n. \forall Y \in \mathbb{R}^m. (\Phi_{0,\leq}(X, Y)),$$

where $\Phi_{0,\leq}$ is a QFF_{\leq} -formula, we can compute in polynomial time an equivalent $\mathfrak{b}\text{-}\Sigma_{\leq}$ -sentence

$$\exists X \in I^n. \forall Y \in I^{n+M}. (\Psi_{0,\leq}(X, Y)).$$

Proof Sketch. The proof combines Lemma 12 with proof ideas similar to those used in the proof of Lemma 10. See [13, Lemma 14] for details. ◀

Lemmas 10 and 13 together yield the inclusion $\exists\forall_{\leq}\mathbb{R} \subseteq \mathfrak{b}\text{-}\exists\forall_{\leq}\mathbb{R}$.

33:10 On the Complexity of the Compact Escape Problem

3.2 Showing $\mathbf{b}\text{-}\exists\forall_{\leq}\mathbb{R} \subseteq \exists\forall_{\leq}\mathbb{R}$

We next establish the inclusion $\mathbf{b}\text{-}\exists\forall_{\leq}\mathbb{R} \subseteq \exists\forall_{\leq}\mathbb{R}$. The key lemma is the following:

► **Lemma 14.** *Let*

$$\exists \varepsilon > 0. (Q_1 X \in \mathbb{R}^n). (Q_2 Y \in \mathbb{R}^m). (\Phi_0(\varepsilon, X, Y))$$

be a sentence over the language \mathcal{L} of matrix size μ . If this sentence holds true, then there exists $\varepsilon > 2^{-\mu^{4\alpha^3\beta(n+1)(m+1)}}$ witnessing the existential quantifier.

Proof. Consider the formula

$$\chi_2(\varepsilon) = (Q_1 X \in \mathbb{R}^n). (Q_2 Y \in \mathbb{R}^m). (\Phi_0(\varepsilon, X, Y)).$$

By Theorem 8 this formula is equivalent to a quantifier-free formula $\chi_0(\varepsilon)$ of size at most $\mu^{2\alpha^3(n+1)(m+1)}$. Let $\chi'_0(\varepsilon)$ be the sentence that results from χ_0 by replacing each atom in $P(\varepsilon) \bowtie 0$ in χ_0 , where P has degree d , with the atom $\varepsilon^d P(1/\varepsilon) \bowtie 0$. Then, evidently, a number $\varepsilon > 0$ satisfies $\chi_0(\varepsilon)$ if and only if $1/\varepsilon$ satisfies $\chi'_0(\varepsilon)$ and vice versa.

By Corollary 9 the sentence $\exists x \in \mathbb{R}. (x > 0 \wedge \chi'_0(x))$ is equivalent to the sentence

$$\exists x \in \mathbb{R}. \left(x > 0 \wedge |x| \leq 2^{\mu^{4\alpha^3\beta(n+1)(m+1)}} \wedge \chi'_0(x) \right).$$

The result follows. ◀

► **Theorem 15.** *Given a $\mathbf{b}\text{-}\Sigma_{2,\leq}$ -sentence*

$$\exists X \in I^m. \forall Y \in I^m. (\Phi_{0,\leq}(X, Y))$$

we can compute in polynomial time an equivalent $\Sigma_{2,\leq}$ -sentence.

Proof Sketch. The proof combines Lemma 14 and Proposition 11 with similar ideas as in the proof of Lemma 10. See [13, Theorem 16] for details. ◀

3.3 Showing $\mathbf{b}\text{-}\exists\forall_{\leq}^{++}\mathbb{R} \subseteq \mathbf{b}\text{-}\exists\forall_{\leq}\mathbb{R}$

Finally we show the inclusion $\mathbf{b}\text{-}\exists\forall_{\leq}^{++}\mathbb{R} \subseteq \mathbf{b}\text{-}\exists\forall_{\leq}\mathbb{R}$.

We will in fact show a stronger but more technical result. Recall that the Hausdorff distance of two non-empty compact subsets K and L of a metric space X is given by

$$d(K, L) = \max\left\{ \sup_{x \in K} d(x, L), \sup_{x \in L} d(x, K) \right\},$$

where, as usual, $d(x, K) = \inf_{y \in K} d(x, y)$. This distance function makes the non-empty compact subsets of a metric space into a metric space $\mathcal{F}(X)$ of its own.

► **Theorem 16.** *Consider a sentence of the form*

$$\exists X \in I^n. \forall Y \in I^m. (\Psi_{0,\leq}(X, Y) \rightarrow \Phi_{0,\leq}(X, Y)),$$

where $\Psi_{0,\leq}(X, Y)$ and $\Phi_{0,\leq}(X, Y)$ are QFF $_{\leq}$ -formulas. Assume that the set-valued function $F(X) = \{Y \in I^m \mid \Psi_{0,\leq}(X, Y)\}$ either maps some $X \in I^n$ to the empty set or is continuous as a map of type $I^n \rightarrow \mathcal{F}(I^m)$. Then we can compute in polynomial time an equivalent $\mathbf{b}\text{-}\Sigma_{2,\leq}$ -sentence.

We prepare the proof of Theorem 16 with three simple observations.

► **Lemma 17.** *Given a sentence of the form*

$$\exists X \in I^n . \forall Y \in I^m . (H(X, Y) > 0),$$

where H is a multivariate polynomial with integer coefficients we can compute in polynomial time an equivalent $\mathfrak{b}\text{-}\Sigma_{2, \leq}$ -sentence.

Proof. The sentence is equivalent to the sentence

$$\exists \varepsilon > 0 . \exists X \in I^n . \forall Y \in I^m . (H(X, Y) \geq \varepsilon).$$

By Lemma 14 this sentence is equivalent to the sentence

$$\exists \varepsilon \in I . \exists X \in I^n . \forall Y \in I^m . \left(\varepsilon \geq 2^{\mu^{4\alpha^3\beta(n+1)(m+1)}} \wedge H(X, Y) \geq \varepsilon \right),$$

where μ is the size of h . Compute in polynomial time an integer N such that $\mu^{4\alpha^3\beta(n+1)(m+1)} \leq 2^N$ and apply Lemma 25 to obtain the result. ◀

► **Lemma 18.** *Let $P \in \mathbb{Z}[X]$ be a polynomial in n variables, encoded by a term T over the signature $\langle \mathbb{Z}, +, \times \rangle$. Then we can compute in polynomial time an integer N (in binary) such that $|P(I^n)| \leq N$.*

Proof. We can view T as a tree whose nodes are elements of the set $\{+, \times\}$ and whose leaves are either variables or constants. Let $c_1, \dots, c_s \in \mathbb{Z}$ denote the integer constants that occur in T . Let $M = \max\{2, |c_1|, \dots, |c_s|\}$.

Let S be the tree which is obtained by substituting M for all leaves in T . Then S encodes a positive integer B . This integer B is clearly an upper bound for the absolute value of P over I^n . By an easy induction argument B is bounded by M^{N_T} , where N_T is the number of nodes of T . The number M^{N_T} can be computed using at most N_T arithmetic operations. Its bitsize is bounded by $N_T\tau$, where τ is a bound on the bitsizes of the numbers c_1, \dots, c_s . ◀

► **Proposition 19.** *Let $\Phi(X)$ be a quantifier-free formula over the language \mathcal{L} whose atoms consist of equalities only. Then we can compute in polynomial time a polynomial $Q \in \mathbb{Z}[X]$ such that $\Phi(X)$ is equivalent to the formula $Q(X) = 0$.*

Proof. Construct a new formula $\Phi'(X)$ that results from $\Phi(X)$ by replacing each atom $P(X) = 0$ in $\Phi(X)$ by the atom $P(X)^2 = 0$.

Now construct a polynomial $Q_{\Phi'}$ by structural induction on Φ' as follows:

1. If $\Phi'(X) \equiv (P(X) = 0)$ then let $Q_{\Phi'} = P$.
2. If $\Phi'(X) \equiv \Psi(X) \vee \omega(X)$ then let $Q_{\Phi'} = Q_{\Psi} \cdot Q_{\omega}$.
3. If $\Phi'(X) \equiv \Psi(X) \wedge \omega(X)$ then let $Q_{\Phi'} = Q_{\Psi} + Q_{\omega}$.

It is easy to see that $Q_{\Phi'}$ can be computed in polynomial time from Φ . It has the desired property by construction. ◀

We are now in a position to prove Theorem 16.

Proof of Theorem 16. The proof is a reduction to Lemma 17.

As a preparation we assign to every QFF $_{\leq}$ -formula Φ a continuous function f_{Φ} such that $\Phi(X)$ holds true if and only if $f_{\Phi}(X) \leq 0$:

1. If $\Phi(X) = (P(X) \leq 0)$ then let $f_{\Phi}(X) = P(X)$.
2. If $\Phi(X) = \Psi(X) \vee \chi(X)$ then let $f_{\Phi}(X) = \min\{f_{\Psi}(X), f_{\chi}(X)\}$.
3. If $\Phi(X) = \Psi(X) \wedge \chi(X)$ then let $f_{\Phi}(X) = \max\{f_{\Psi}(X), f_{\chi}(X)\}$.

33:12 On the Complexity of the Compact Escape Problem

Now assume we are given a sentence

$$\exists X \in I^n. \forall Y \in I^m. (\Psi(X, Y) \rightarrow \Phi(X, Y)) \quad (5)$$

as above. The negation of this sentence is equivalent to the sentence

$$\forall X \in I^n. \exists Y \in I^m. (\Psi(X, Y) \wedge f_\Phi(X, Y) > 0). \quad (6)$$

Let us for now assume that the set $K(X) = \{Y \in I^m \mid \Psi(X, Y)\}$ is non-empty for all $X \in I^n$. Then by assumption this set depends continuously on X in the Hausdorff metric. It follows by elementary calculus that the function $h(X) = \max_{Y \in K(X)} f_\Phi(X, Y)$ is well-defined and continuous.

We further have, by compactness of I^n , that the function $h(X)$ attains its minimum in I^n . By definition of f_Φ , the sentence (6) holds true if and only if $\min_{x \in I^n} h(x) > 0$ if and only if there exists $\varepsilon > 0$ such that $\min_{x \in I^n} h(x) > \varepsilon$. Thus, the sentence (6) is equivalent to the sentence

$$\exists \varepsilon > 0. \forall X \in I^n. \exists Y \in I^m. (\Psi(X, Y) \wedge f_\Phi(X, Y) > \varepsilon).$$

So far we have proved this equivalence under the assumption that the compact set $K(X) = \{Y \in I^m \mid \Psi(X, Y)\}$ is non-empty for all X . But if the set $K(X)$ is empty for some X then both (6) and the above sentence are false, so that the two sentences are certainly equivalent.

Let $\chi(X, Y)$ be the formula that results from Φ by swapping all occurrences of \vee and \wedge and by replacing all atoms $P(X, Y) \leq 0$ in Φ by the atom $P(X, Y) > \varepsilon$. One easily checks that the above sentence is further equivalent to the sentence

$$\exists \varepsilon > 0. \forall X \in I^n. \exists Y \in I^m. (\Psi(X, Y) \wedge \chi(X, Y)).$$

It follows from 10 that there exists a witness ε for the existential quantifier with $\varepsilon > 2^{-\mu^{4\alpha^3\beta(n+1)(m+1)}}$. We can compute in polynomial time an integer N such that we have $\mu^{4\alpha^3\beta(n+1)(m+1)} \leq 2^N$. Consider the formula $\chi(X, Y)$. By Lemma 18 we can compute in polynomial time an integer L such that $|P(X, Y)| \leq L$ for all $(X, Y) \in I^n \times I^m$. We can hence replace each atom $P(X, Y) > 0$ in $\chi(X, Y)$ with the equivalent formula

$$\exists u \in [-L, L]. \exists v \in [-2^{2^N}, 2^{2^N}]. (P(X, Y) = u^2 \wedge uv = 1),$$

where u and v are fresh variables. By Proposition 19 the formula $\chi(X, Y)$ is equivalent to a formula of the form

$$\exists U \in [-L, L]^s. \exists V \in [-2^{2^N}, 2^{2^N}]^s. (Q(X, Y, U, V) = 0)$$

where Q is computable in polynomial time from $\chi(X, Y)$ and s is the number of atoms in $\chi(X, Y)$.

Now, consider the formula $\Psi(X, Y)$. By Lemma 18 we can compute in polynomial time an integer M such that for all atoms $P(X, Y) \leq 0$ in $\Psi(Y)$ the polynomial P satisfies $|P(X, Y)| \leq M$ for all $(X, Y) \in I^n \times I^m$. The atom is hence equivalent to $\exists w \in [-M, M]. P(X, Y) = -w^2$, where w is a fresh variable. Again by Proposition 19, letting t denote the number of atoms in $\Psi(X, Y)$ we can hence compute in polynomial time a formula $\exists W \in [-M, M]^t. R(X, Y, W) = 0$, which is equivalent to $\Psi(X, Y)$.

In total the sentence (6) is equivalent to the sentence

$$\forall X \in I^n. \exists Y \in I^m. \exists U \in [-L, L]^s. \exists V \in [-2^{2^N}, 2^{2^N}]^s. \exists W \in [-M, M]^t. (R(X, Y, W) + Q(X, Y, U, V) = 0).$$

In the above we have used that the functions R and Q admit only non-negative values by construction. We may assume that $2^{2^N} \geq \max\{L, M\}$. Arguing as in the proof of Lemma 25 in Appendix A, we can introduce auxiliary variables $B \in I^{N+1}$ to obtain an equivalent sentence

$$\forall X \in I^n. \exists Y \in I^m. \exists U \in I^s. \exists V \in I^s. \exists W \in I^t. \exists B \in I^{N+1}. (H(X, Y, U, V, W, B) = 0)$$

which is computable in polynomial time from our original sentence (5).

The sentence (5) is hence equivalent to the sentence

$$\exists X \in I^n. \forall Y \in I^m. \forall U \in I^s. \forall V \in I^s. \forall W \in I^t. \forall B \in I^{N+1}. (H(X, Y, U, V, W, B) > 0).$$

Again, we have used that H only admits non-negative values by construction. The result now follows from Lemma 17. \blacktriangleleft

The inclusion $\mathbf{b}\text{-}\exists\forall_{\leq}^+\mathbb{R} \subseteq \mathbf{b}\text{-}\exists\forall_{\leq}\mathbb{R}$ follows from the special case of Theorem 16 where the formula $\Psi_{0, \leq}(Y)$ does not depend on X .

Theorem 16, in its general form, finally allows us to prove that the complexity class $\exists\forall_{\leq}\mathbb{R}$ is robust under different encodings of polynomials.

► Theorem 20. *Given a \mathcal{C} -sentence, where $\mathcal{C} \in \{\Sigma_{2, \leq}, \mathbf{b}\text{-}\Sigma_{2, \leq}, \mathbf{b}\text{-}\Sigma_{2, \leq}^p\}$ we can compute in polynomial time an equivalent \mathcal{C} -sentence whose atoms involve polynomials of degree at most four. In particular we can compute in polynomial time a sentence whose atoms involve polynomials encoded as in (1).*

The proof of Theorem 20 requires the following proposition, which is easily established using elementary calculus:

► Proposition 21. *Let X and Y be metric spaces.*

1. *Let $F: X \rightarrow \mathcal{F}(Y)$ and $G: X \rightarrow \mathcal{F}(Z)$ be continuous with respect to the Hausdorff metric. Then the map*

$$H: X \rightarrow \mathcal{F}(Y) \times \mathcal{F}(Z), H(x) = F(x) \times G(x)$$

is continuous with respect to the Hausdorff metric as well.

2. *Let $F: X \rightarrow \mathcal{F}(Y)$ be continuous with respect to the Hausdorff metric. Let $f: Y \rightarrow Z$ be a continuous function. Then the function*

$$H: X \rightarrow \mathcal{F}(Y \times Z), H(x) = F(x) \times f(F(x))$$

is continuous with respect to the Hausdorff metric.

Proof of Theorem 20. We prove the result for $\mathbf{b}\text{-}\Sigma_{2, \leq}$ -sentences. The result for $\Sigma_{2, \leq}$ -sentences follows by applying the reductions from Lemmas 10 and 12, bounding the degrees of the atoms of the resulting $\mathbf{b}\text{-}\Sigma_{2, \leq}$ -sentence, and translating back to a $\Sigma_{2, \leq}$ -sentence using Theorem 15. By inspecting the proof of Theorem 15 we observe that the degree does not increase by this translation, since we only add new constraints, all of which involve polynomials of degree at most 2. The result for $\mathbf{b}\text{-}\Sigma_{2, \leq}^+$ -sentences is implicitly contained in the below proof.

To a term T over the signature $\langle \mathbb{Z}, +, \times \rangle$ we assign a variable z_T and a formula η_T , where η_T is inductively defined as follows:

33:14 On the Complexity of the Compact Escape Problem

1. If T is a variable x_j then $\eta_T = \langle z_T = x_j \rangle$.
2. If T is a constant c then $\eta_T = \langle z_T = c \rangle$.
3. If T is of the form $U \times V$, then $\eta_T = \langle \eta_U \wedge \eta_V \wedge z_T = z_U \times z_V \rangle$
4. If T is of the form $U + V$, then $\eta_T = \langle \eta_U \wedge \eta_V \wedge z_T = z_U + z_V \rangle$.

The formula η_T is computable in polynomial time from T . Its atoms have degree at most two.

Let $P(X, Y) \leq 0$ be an atom in $\Phi_{\leq}(X, Y)$, where P is encoded by a term T . Let η_T be the formula associated with T as above. Then the formula $P(X, Y) \leq 0$ is equivalent to the formula $\forall Z. (\eta_T(X, Y, Z) \rightarrow z_T \leq 0)$.

More generally, the sentence $\exists X \in I^n. \forall Y \in I^m. \Phi_{\leq}(X, Y)$ is equivalent to the sentence

$$\exists X \in I^n. \forall Y \in I^m. \forall Z \in \mathbb{R}^M. \left(\eta_{T_1}(X, Y, Z) \wedge \cdots \wedge \eta_{T_s}(X, Y, Z) \rightarrow \widehat{\Phi}_{\leq}(Z) \right),$$

where T_1, \dots, T_s are the term representations of the atoms in $\Phi_{\leq}(X, Y)$ and $\widehat{\Phi}_{\leq}(Z)$ is obtained from $\Phi_{\leq}(X, Y)$ by substituting each atom $P(X, Y) \leq 0$ with term representation T_j by the atom $z_{T_j} \leq 0$.

We can further compute in polynomial time an integer N in binary such that the above sentence is equivalent to

$$\exists X \in I^n. \forall Y \in I^m. \forall Z \in [-N, N]^M. \left(\eta_{T_1}(X, Y, Z) \wedge \cdots \wedge \eta_{T_s}(X, Y, Z) \rightarrow \widehat{\Phi}_{\leq}(Z) \right),$$

By the proof of Lemma 25 we can have Z range over $[-1, 1]^M$ up to introducing further auxiliary variables and adding a conjunction of quadratic polynomial equations to the formula $\widehat{\Phi}$. For notational convenience, let us simply assume that the sentence is equivalent to

$$\exists X \in I^n. \forall Y \in I^m. \forall Z \in I^M. \left(\eta_{T_1}(X, Y, Z) \wedge \cdots \wedge \eta_{T_s}(X, Y, Z) \rightarrow \widehat{\Phi}_{\leq}(Z) \right).$$

This sentence involves polynomials of degree at most 2.

Let us write $\eta(X, Y, Z) = \bigwedge_{j=1}^s \eta_{T_j}(X, Y, Z)$. It remains to show that the set

$$\{(Y, Z) \in I^m \times I^M \mid \eta(X, Y, Z)\}$$

depends continuously on X in the Hausdorff metric. It then follows from Theorem 16 that we can compute in polynomial time an equivalent $\Sigma_{2, \leq}$ -sentence. By an inspection of the proof of Theorem 16, the degree of the atoms is at most doubled in this new sentence.

Now, The formula η is a conjunction of atoms of the form $z_j = x_k$, $z_j = y_k$, $z_j = c$, $z_j = z_k + z_\ell$, or $z_j = z_k \times z_\ell$.

We prove the result by structural induction, using Proposition 21. For a formula $\eta(X, Y, Z)$ with $n + m + s$ free variables (X, Y, Z) write $F_\eta: I^n \rightarrow \mathcal{F}(I^{m+s})$ for the map that sends $X \in I^n$ to the set $\{(Y, Z) \in I^m \times I^s \mid \eta(X, Y, Z)\}$.

If $\eta(X, Y, z)$ is of the form $z = x_k$, $z = y_k$, or $z = c$ then the function F_η is easily seen to be continuous.

If $\eta(X, Y, z_1, \dots, z_s) = \nu(X, Y, z_1, \dots, z_{s-1}) \wedge \mu(X, Y, z_s)$ where $\mu(X, Y, z_s)$ is of the form $z_s = x_k$, $z_s = y_k$, or $z_s = c$ then

$$F_\eta(X) = F_\nu(X) \times \{z_s \in \mathbb{R} \mid \mu(X, Y, z_s)\}.$$

Continuity of F_η follows from the first part of Proposition 21.

If $\eta(X, Y, z_1, \dots, z_s) = \nu(X, Y, z_1, \dots, z_{s-1}) \wedge \mu(X, Y, z_j, z_k, z_s)$ where $\mu(X, Y, z_j, z_k, z_s)$ is of the form $z_s = z_j \square z_k$ with $\square \in \{+, \times\}$, then

$$F_\eta(X) = F_\nu(X) \times f(F_\nu(X)),$$

where $f(Y, z_1, \dots, z_{s-1}) = z_j \square z_k$. Continuity of F_η follows from the second part of Proposition 21. ◀

4 The complexity of deciding the Compact Escape Problem

We show that CEP is complete for the complexity class $\exists\forall_{\leq}\mathbb{R}$. Formally this is achieved by locating CEP between the complexity classes $\mathbf{b}\text{-}\exists\forall_{\leq}\mathbb{R}$ and $\mathbf{b}\text{-}\exists\forall_{\leq}^{++}\mathbb{R}$ and applying Theorem 3.

Let us first show that CEP is $\exists\forall_{\leq}\mathbb{R}$ -hard. As a preparation we need to construct in polynomial time an arbitrary finite number of irrational rotations with independent angles:

► **Lemma 22.** *Given $n \in \mathbb{N}$ in unary we can compute in polynomial time a set of points $q_1, \dots, q_n \in \mathbb{T}^1 \subseteq \mathbb{C}$ with rational real and imaginary part such that the only integer solution $(e_1, \dots, e_n) \in \mathbb{Z}^n$ to the equation $q_1^{e_1} \cdots q_n^{e_n} = 1$ is the zero vector.*

Proof. See [13, Lemma 20]. ◀

► **Theorem 23.** *The Compact Escape Problem is $\exists\forall_{\leq}\mathbb{R}$ -hard.*

Proof. By Theorem 3 the decision problem for $\mathbf{b}\text{-}\Sigma$ -sentences is $\exists\forall_{\leq}\mathbb{R}$ -complete. It hence suffices to reduce this problem to CEP.

Thus, given a $\mathbf{b}\text{-}\Sigma_{2,\leq}$ -sentence $\Psi_{2,\leq} = \exists x \in I^n. \forall y \in I^m. (\Phi_{0,\leq}(x, y))$ we compute in polynomial time a compact set K and a rational matrix $A \in \mathbb{Q}^{(n+2m) \times (n+2m)}$ such that there exists a point $x \in K$ with $A^k x \in K$ for all $n \in \mathbb{N}$ if and only if $\Psi_{2,\leq}$ holds true.

By Theorem 20 we may assume that all polynomials that occur in $\Psi_{2,\leq}$ have degree at most 4.

Consider the compact set

$$K = \{(x, u_1, v_1, \dots, u_m, v_m) \in I^n \times I^{2m} \mid u_j^2 + v_j^2 = 1, \Phi_{0,\leq}(x, u_1, \dots, u_m)\}.$$

Use Lemma 22 to compute rational numbers $a_1, \dots, a_m, b_1, \dots, b_m \in \mathbb{Q}$ such that the numbers $a_j + ib_j$ do not admit any non-trivial integer multiplicative relations. Denote by I_n the $(n \times n)$ -identity matrix. Let $R \in \mathbb{Q}^{2m \times 2m}$ be the matrix corresponding to the linear transform which sends a vector $(x_1, y_1, \dots, x_m, y_m) \in \mathbb{Q}^{2m}$ to the vector

$$(a_1 x_1 - b_1 y_1, b_1 x_1 + a_1 y_1, \dots, a_m x_m - b_m y_m, b_m x_m + a_m y_m).$$

Let $A \in \mathbb{Q}^{(n+2m) \times (n+2m)}$ be defined as follows:

$$A = \begin{pmatrix} I_n & \\ & R \end{pmatrix}.$$

Then for all $x \in K$ we have by Theorem 6

$$\overline{\mathcal{O}_A(x)} = \{x\} \times \{(u_1, v_1, \dots, u_m, v_m) \in I^{2m} \mid u_j^2 + v_j^2 = 1\}.$$

It follows that $\overline{\mathcal{O}_A(x)} \subseteq K$ if and only if $\Phi_{0,\leq}(x, u_1, \dots, u_m)$ holds true for all $u_1, \dots, u_m \in I^m$.

Thus, the instance (A, K) of CEP is a negative instance if and only if the sentence $\Psi_{2,\leq}$ holds true. We can compute (A, K) in polynomial time from $\Psi_{2,\leq}$. This is almost immediately obvious, except that the polynomial inequalities that represent K must be encoded as lists of coefficients, while the polynomial inequalities in $\Psi_{2,\leq}$ are given as terms over the signature $\langle \mathbb{Z}, +, \times \rangle$. But since the polynomials that occur in $\Psi_{2,\leq}$ have degree at most 4 we can efficiently compute a list of coefficients from the term representations. ◀

Conversely, we have:

► **Theorem 24.** *The Compact Escape Problem is contained in $\exists\forall_{\leq}\mathbb{R}$.*

Proof Sketch. The full proof is given in Appendix B. We will only briefly sketch the proof idea here.

Suppose we are given a matrix $A \in \mathbb{Q}^{n \times n}$ with rational entries and a family of polynomials \mathcal{P} together with a negation-free propositional formula which encodes a compact set $K \subseteq \mathbb{R}^n$. We can compute in polynomial time from this data a QFF $_{\leq}$ -formula Φ which encodes K . We will show that the existence of a point in K that is trapped under A is expressible as a $\mathbf{b}\text{-}\Sigma_{2,\leq}^{++}$ -sentence. Together with Theorem 3 this yields the result. Let us assume for the sake of simplicity that A is diagonalisable over the complex numbers. The general case employs the Jordan normal form. It is not more difficult but requires more cumbersome notation.

We compute the complex eigenvalues $\lambda_1, \dots, \lambda_m, \lambda_{m+1}, \dots, \lambda_{m+b}, \lambda_{m+b+1}, \dots, \lambda_{m+b+s}$ of A , counted with multiplicity. The eigenvalues are labelled such that $\lambda_1, \dots, \lambda_m$ have modulus 1, such that $\lambda_{m+1}, \dots, \lambda_{m+b}$ have modulus strictly greater than 1, and such that $\lambda_{m+b+1}, \dots, \lambda_{m+b+s}$ have modulus strictly smaller than 1. Using [7] we can compute in polynomial time base change matrices Q and Q^{-1} such that $D = Q^{-1}AQ$ is a diagonal matrix.

Let $x \in K$ be a starting point. If the complex vector $Q^{-1}x$ has a non-zero component $(Q^{-1}x)_j$ with $m+1 \leq j \leq m+b$ then the orbit of x under A is unbounded, and hence forced to leave the bounded set K .

Now assume that $(Q^{-1}x)_j = 0$ for all $m+1 \leq j \leq m+b$. All components $(Q^{-1}x)_j$ with $j \geq m+b+1$ converge to zero under the iteration of A in the sense that the sequence $(Q^{-1}(A^k x))_j$ converges to zero as $k \rightarrow \infty$. It follows that the closure of the orbit of x under A is equal to the range of the semialgebraic function

$$f(x, z) = Q \operatorname{diag}(z_1, \dots, z_m, 0, \dots, 0) Q^{-1}x,$$

where z_1, \dots, z_m range over the closure of the sequence $(\lambda_1^k, \dots, \lambda_m^k)_k$ in the torus \mathbb{T}^m . By Theorem 6 the closure of this sequence is an algebraic subset of \mathbb{T}^m , cut out by the integer multiplicative relations between the eigenvalues $\lambda_1, \dots, \lambda_m$. By Theorem 7 a QFF $_{\leq}$ -formula $\Psi(Z)$ encoding this algebraic set, up to identifying \mathbb{T}^m with a subset of the real hypercube $I^{2m} \subseteq \mathbb{R}^{2m}$.

It follows that we can express the existence of a trapped point by the following “informal” sentence:

$$\begin{aligned} &\exists X \in I^n. \forall Z \in I^{2n}. \\ &(\Psi(Z) \rightarrow (X \in K \wedge ((Q^{-1}X)_{m+1} = 0 \wedge \dots \wedge (Q^{-1}X)_{m+b} = 0) \wedge f(X, Z) \in K)). \end{aligned}$$

Thanks to the polytime computability of Q and Q^{-1} we can compute in polynomial time formulas that express the relations $(Q^{-1}X)_j = 0$ for $j = m+1, \dots, m+b$, and $f(X, Z) \in K$. This allows us to compute in polynomial time a $\mathbf{b}\text{-}\Sigma_{\leq}^{++}$ -sentence which is equivalent to the above “informal” sentence. ◀

References

- 1 Rajeev Alur. *Principles of Cyber-Physical Systems*. MIT Press, 2015.
- 2 Andrea Bacciotti and Luisa Mazzi. Stability of dynamical polysystems via families of Lyapunov functions. *J. Nonlinear Analysis*, 67:2167–2179, 2007.
- 3 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2006.
- 4 Saugata Basu and Marie-Françoise Roy. Bounding the radii of balls meeting every connected component of semi-algebraic sets. *Journal of Symbolic Computation*, 45(12):1270–1279, 2010.

- 5 Vincent D. Blondel and John N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000. doi:10.1016/S0005-1098(00)00050-9.
- 6 Mark Braverman. Termination of integer linear programs. In *CAV’06*, volume 4144 of *LNCS*. Springer, 2006.
- 7 Jin-Yi Cai. Computing Jordan normal forms exactly for commuting matrices in polynomial time. *International Journal of Foundations of Computer Science*, 5(3/4):293–302, 1994. doi:10.1142/S0129054194000165.
- 8 Jin-Yi Cai, Richard J. Lipton, and Yechezkel Zalcstein. The complexity of the A B C problem. *J. Computing*, 29(6), 2000.
- 9 John Canny. Some algebraic and geometric computations in PSPACE. In *STOC’88*, pages 460–467. ACM, 1988.
- 10 Eugênio B. Castelan and Jean-Claude Hennet. On invariant polyhedra of continuous-time linear systems. *IEEE Transactions on Automatic Control*, 38(11):1680–85, 1993.
- 11 Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1993.
- 12 George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages*, pages 134–183. Springer Berlin Heidelberg, 1975.
- 13 Julian D’Costa, Engel Lefaucheux, Eike Neumann, Joël Ouaknine, and James Worrell. On the complexity of the escape problem for linear dynamical systems over compact semialgebraic sets, 2021. arXiv:2107.02060.
- 14 Dima Grigoriev. Complexity of deciding Tarski algebra. *J. Symbolic Computation*, 5(1–2):65–108, 1988.
- 15 Dima Grigoriev and Nicolai Vorobjov. Solving systems of polynomial inequalities in subexponential time. *J. Symbolic Computation*, 5:37–64, 1988.
- 16 Joos Heintz, Marie-Françoise Roy, and Pablo Solernó. Sur la complexité du principe de Tarski-Seidenberg. *Bull. Soc. Math. France*, 118(1):101–126, 1990.
- 17 David W. Masser. *Linear relations on algebraic groups*, page 248–262. Cambridge University Press, 1988.
- 18 Eike Neumann, Joël Ouaknine, and James Worrell. On ranking function synthesis and termination for polynomial programs. In *CONCUR’20*, volume 171, pages 15:1–15:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 19 Joël Ouaknine and James Worrell. *Positivity Problems for Low-Order Linear Recurrence Sequences*, page 366–379. Society for Industrial and Applied Mathematics, USA, 2014.
- 20 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. i-iii. *J. Symbolic Computation*, 13(3):255–352, 1992.
- 21 Sriram Sankaranarayanan, Thao Dang, and Franjo Ivancic. A policy iteration technique for time elapse over template polyhedra. In *HSCC’08*, volume 4981 of *LNCS*. Springer, 2008.
- 22 Marcus Schaefer and Daniel Štefankovič. Fixed Points, Nash Equilibria, and the Existential Theory of the Reals. *Theory Computing Systems*, 60(2):172–193, 2017.
- 23 Shashi M. Srivastava. *A course on Mathematical Logic*. Springer, 2008.
- 24 Ashish Tiwari. Termination of linear programs. In *CAV’04*, volume 3114 of *LNCS*. Springer, 2004.
- 25 Dirk van Dalen. *Logic and Structure*. Springer Berlin Heidelberg, fourth edition, 2004.
- 26 Nicolai Vorobjov. Bounds of real roots of a system of algebraic equations. *Zap. Nauchn. Sem. LOMI*, 137:7–19, 1984. (in Russian).

A Removing doubly exponential bounds

► **Lemma 25.** *Given an integer N in unary and a sentence*

$$(Q_1 X_1).(Q_2 X_2).\dots(Q_s X_s).\Phi_0(X_1, \dots, X_s),$$

we can in polynomial time in the size of the sentence and N compute a sentence

$$\exists B \in [-1, 1]^{N+1}.(Q_1 X_1; |X_1| \leq 1).(Q_2 X_2; |X_2| \leq 1).\dots(Q_s X_s; |X_s| \leq 1). \\ \Psi_0(B, X_1, \dots, X_s)$$

which is equivalent to the sentence

$$(Q_1 X_1; |X_1| \leq 2^{2^N}).(Q_2 X_2; |X_2| \leq 2^{2^N}).\dots(Q_s X_s; |X_s| \leq 2^{2^N}).\Phi_0(X_1, \dots, X_s).$$

Here, the notation $(Q_j; |X_j| \leq c)$ indicates that the quantifier is restricted to the set

$$\{X_j \in \mathbb{R}^{n_j} \mid |X_{j,1}| \leq c, \dots, |X_{j,n_j}| \leq c\}.$$

Further, if Φ_0 is a QFF $_{\leq}$ -formula then so is Ψ_0 .

Proof. Introduce fresh variables b_0, \dots, b_N . Let Ψ'_0 be the formula that results from Φ_0 by replacing each atom

$$P(X_1, \dots, X_s) \bowtie 0$$

in Φ_0 , where $\bowtie \in \{\leq, <, =\}$, by the atom

$$b_N^{d_P} \cdot P(X_1/b_N, \dots, X_s/b_N) \bowtie 0,$$

where d_P is the total degree of P . Let Ψ_0 be the formula

$$\Psi'_0 \wedge 2b_0 = 1 \wedge b_1 = b_0^2 \cdots \wedge b_N = b_{N-1}^2. \quad \blacktriangleleft$$

B Proof of Theorem 24

We start with a technical lemma:

► **Lemma 26.** *Let $A \in \mathbb{R}^{n \times n}$ be a real matrix. Denote by*

$$\lambda_1, \dots, \lambda_m, \lambda_{m+1}, \dots, \lambda_{m+b}, \lambda_{m+b+1}, \dots, \lambda_{m+b+s}$$

the complex eigenvalues of A , counted with geometric multiplicity. Let $\lambda_1, \dots, \lambda_m$ have modulus 1. Let $\lambda_{m+1}, \dots, \lambda_{m+b}$ have modulus strictly greater than 1. Let $\lambda_{m+b+1}, \dots, \lambda_{m+b+s}$ have modulus strictly smaller than 1. Fix a Jordan basis $v_{j,k}$ of \mathbb{C}^n where $v_{j,1}$ is an eigenvector of λ_j and $(A - \lambda_j I)v_{j,k} = v_{j,k-1}$ for all $k > 1$.

Let B denote the span of the vectors $v_{j,k}$ with $m+1 \leq j \leq m+b$ and the vectors $v_{j,k}$ with $1 \leq j \leq m$ and $k > 1$.

Let C denote the span of the vectors $v_{j,k}$ with $m+b+1 \leq j \leq m+b$.

Let Q be the matrix that sends the standard basis of \mathbb{C}^n to the basis

$$v_{1,1}, \dots, v_{m,1},$$

$$v_{1,2}, \dots, v_{1,t_1}, \dots, v_{m,2}, \dots, v_{m,t_m},$$

$$v_{m+1,1}, \dots, v_{v_{m+1},t_{m+1}}, \dots, v_{m+b+1,1}, \dots, v_{m+b+1,t_{m+b+1}}.$$

Let

$$f: \mathbb{R}^n \times \mathbb{T}^m \rightarrow \mathbb{C}^n, f(x, z) = Q \begin{pmatrix} z_1 & & & & & \\ & \ddots & & & & \\ & & z_m & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} Q^{-1} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

Let $S \subseteq \mathbb{T}^m$ be the closure of the set $\{(\lambda_1^k, \dots, \lambda_m^k) \mid k \in \mathbb{N}\}$ in \mathbb{T}^m .

Let $K \subseteq \mathbb{R}^n$ be a compact set. Let $x \in K$. Then for all $k \in \mathbb{N}$ we have $A^k x \in K$ if and only if both of the following two conditions are satisfied:

1. Let $N = m + (t_1 - 1) + \dots + (t_m - 1) + t_{m+1} + \dots + t_{m+b}$. For all $m < j \leq N$ we have $(Q^{-1}x)_j = 0$.
2. $f(x, S) \subseteq K$.

Proof. Let $x \in K$.

Assume that $A^k x \in K$ for all $k \in \mathbb{N}$. Let $J = Q^{-1}AQ$. Let us again write $N = m + (t_1 - 1) + \dots + (t_m - 1) + t_{m+1} + \dots + t_{m+b}$. If there exists $m < j \leq N$ such that $(Q^{-1}x)_j \neq 0$ then $Q^{-1}x$ has a non-zero component in a generalised eigenspace of A which corresponds to an eigenvalue of modulus strictly greater than 1 or it has a non-zero component in a generalised eigenspace of A corresponding to an eigenvalue of modulus 1 which is not an eigenspace. In both cases the absolute value of $A^k x = QJ^k(Q^{-1}x)$ is unbounded as $k \rightarrow \infty$. Since K is assumed to be bounded it follows that $A^k x$ leaves K after finitely many steps.

Now, assume that $(Q^{-1}x)_j = 0$ for all $m < j \leq N$. We claim that $f(x, S)$ is the set of accumulation points of the orbit of x under A . The result then follows immediately.

First, observe that we have by construction

$$A = Q \begin{pmatrix} \lambda_1 & & & & & \\ & \ddots & & & & \\ & & \lambda_m & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \\ & & & & & R \end{pmatrix} Q^{-1}$$

where R is an $(s \times s)$ -matrix with $|R^k| \rightarrow 0$ as $k \rightarrow \infty$.

Now, let $z \in S$. We claim that $f(x, z)$ is an accumulation point of the sequence $(A^k x)_{k \in \mathbb{N}}$. Let $\varepsilon > 0$. By Theorem 6 there exist infinitely many $k \in \mathbb{N}$ such that $|\lambda_j^k - z_j| < \varepsilon/2$. For all sufficiently large n we have $|R^k| < \varepsilon/2$. It follows that for each such k we have $|(A^k x) - f(x, z)| < \varepsilon$. Thus, $f(x, z)$ is an accumulation point of the sequence $(A^k x)_k$.

Conversely, let $y \in K$ be an accumulation point of the sequence $(A^k x)_k$. Let $(n_k)_k$ be a sequence of natural numbers such that the sequence $(A^{n_k} x)_k$ converges to y . Since the torus \mathbb{T}^m is compact, the sequence $(\lambda_1^{n_k}, \dots, \lambda_m^{n_k})_k$ has a convergent subsequence. Thus, let $(k_{j_\ell})_\ell$ denote a subsequence of $(k_j)_j$ such that the sequence $(\lambda_1^{k_{j_\ell}}, \dots, \lambda_m^{k_{j_\ell}})_\ell$ converges to a limit $z = (z_1, \dots, z_m) \in \mathbb{T}^m$. Then the sequence $(A^{k_{j_\ell}} x)_\ell$ converges to both $f(x, z)$ and y . It follows that $y = f(x, z)$. ◀

33:20 On the Complexity of the Compact Escape Problem

Now, let us prove Theorem 24.

By Theorem 3 the decision problems for $\mathbf{b}\text{-}\Sigma_{2,\leq}^{++}$ -sentences is contained in $\exists\forall_{\leq}\mathbb{R}$. We reduce the Compact Escape Problem to this problem.

Suppose we are given a matrix $A \in \mathbb{Q}^{n \times n}$ with rational entries, a family of polynomials \mathcal{P} in n free variables, represented in the standard encoding, and a negation-free propositional formula $\Phi(X)$ over atoms of the form $P \leq 0$, where $P \in \mathcal{P}$. We can convert the standard encodings of the polynomials $P \in \mathcal{P}$ into terms over the signature $\langle \mathbb{Z}, +, \times \rangle$ in polynomial time. We can hence convert the formula $\Phi(X)$ into a QFF_{\leq} -formula in polynomial time. By very slight abuse of notation, let us denote this QFF_{\leq} -formula by $\Phi(X)$ as well. Let $K \subseteq \mathbb{R}^n$ denote the set encoded by $\Phi(X)$.

By [7] we can compute in polynomial time the complex eigenvalues of A

$$\lambda_1, \dots, \lambda_m, \lambda_{m+1}, \dots, \lambda_{m+b}, \lambda_{m+b+1}, \dots, \lambda_{m+b+s}$$

and the matrices Q and Q^{-1} as in Lemma 26. We can further compute the real and imaginary parts of the eigenvalues $\lambda_1, \dots, \lambda_{m+b+s}$ in polynomial time. More precisely, letting $\alpha_j = \text{Re}(\lambda_j)$ denote the real part of λ_j , and $\beta_j = \text{Im}(\lambda_j)$ the imaginary part, we can compute in polynomial time:

1. Univariate polynomials with integer coefficients $h_1, \dots, h_{m+b+s}, g_1, \dots, g_{m+b+s}$, such that $h_j(\alpha_j) = g_j(\beta_j) = 0$ for all $j = 1, \dots, m+b+s$.
2. Rational numbers $a_1, b_1, c_1, d_1, \dots, a_{m+b+s}, b_{m+b+s}, c_{m+b+s}, d_{m+b+s}$, such that α_j is the unique root of h_j in the real interval $[a_j, b_j]$ and β_j is the unique root of g_j in the real interval $[c_j, d_j]$.
3. For $j = 1, \dots, n$ and $k = 1, \dots, n$ bivariate polynomials $L_{0,j,k} \in \mathbb{Q}[u, v]$, $L_{1,j,k} \in \mathbb{Q}[u, v]$, and indexes $\ell_{j,k} \in \{1, \dots, m+b+s\}$ such that the matrix Q at row j and column k is given by the complex algebraic number $L_{0,j,k}(\alpha_{\ell_{j,k}}, \beta_{\ell_{j,k}}) + iL_{1,j,k}(\alpha_{\ell_{j,k}}, \beta_{\ell_{j,k}})$.
4. For $j = 1, \dots, n$ and $k = 1, \dots, n$ bivariate polynomials $R_{0,j,k} \in \mathbb{Q}[u, v]$, $R_{1,j,k} \in \mathbb{Q}[u, v]$, and indexes $r_{j,k} \in \{1, \dots, m+b+s\}$ such that the matrix R^{-1} at row j and column k is given by the complex algebraic number $R_{0,j,k}(\alpha_{r_{j,k}}, \beta_{r_{j,k}}) + iR_{1,j,k}(\alpha_{r_{j,k}}, \beta_{r_{j,k}})$.

By Theorem 7 we can compute in polynomial time a finite set $\gamma_1, \dots, \gamma_s \in \mathbb{Z}^m$ of generators of the free abelian group of integer multiplicative relations between the complex eigenvalues $\lambda_1, \dots, \lambda_m$. The size of the integer entries of $\gamma_1, \dots, \gamma_s$ – and not just their bitsize – is bounded polynomially in the size of the input. It follows that we can compute in polynomial time a QFF_{\leq} -formula $\Psi(C, D)$ with $2m$ free variables that expresses for two given real vectors $C \in \mathbb{R}^n$, $D \in \mathbb{R}^n$ that the complex vector $C + iD$ is contained in the set

$$S = \{(z_1, \dots, z_m) \in \mathbb{T}^m \mid (z_1, \dots, z_m)^{\gamma_j} = 1, j = 1, \dots, s\}.$$

By Theorem 6 the set S is equal to the closure of the set $\{(\lambda_1^k, \dots, \lambda_m^k) \mid k \in \mathbb{N}\}$.

Let $f: \mathbb{R}^n \times \mathbb{T}^m \rightarrow \mathbb{C}^n$ be defined as in Lemma 26, i.e.,

$$f(x, z) = Q \text{diag}(z_1, \dots, z_m, 0, \dots, 0) Q^{-1} x.$$

Since we can compute the matrices Q and Q^{-1} in polynomial time as above, we can compute in polynomial time polynomials $F_{k,j} \in \mathbb{Q}[U, V][C, D]$ for $k = 1, \dots, n$, $j = 1, \dots, n$, where U and V are vectors of $m+b+s$ variables, such that

$$\text{Re } f(X, C + iD) = \left(\sum_{j=1}^n F_{1,j}(\vec{\alpha}, \vec{\beta})(C, D) \cdot X_j, \dots, \sum_{j=1}^n F_{n,j}(\vec{\alpha}, \vec{\beta})(C, D) \cdot X_j \right). \quad (7)$$

Note that the result is a polynomial with real algebraic coefficients. More precisely, the right hand side of the above equation is an element of the ring

$$\mathbb{Q}[\alpha_1, \dots, \alpha_{m+b+s}, \beta_1, \dots, \beta_{m+b+s}][X, C, D].$$

Define $N = m + (t_1 - 1) + \dots + (t_m - 1) + t_{m+1} + \dots + t_{m+b}$ as in Lemma 26. By Lemma 26 the existence of a point in K that is trapped under A is equivalent to the “informal” sentence

$$\begin{aligned} \exists X \in I^n. \forall Y \in \mathbb{T}. \\ (Y \in S \rightarrow (X \in K \wedge ((Q^{-1}X)_{m+1} = 0 \wedge \dots \wedge (Q^{-1}X)_N = 0) \wedge f(X, Y) \in K)). \end{aligned} \quad (8)$$

We construct in polynomial time from A and Φ a $\mathbf{b}\text{-}\Sigma_{\leq}^{++}$ -sentence

$$\begin{aligned} \exists U \in I^{m+b+s}. \exists V \in I^{m+b+s}. \exists X \in I^n. \forall C \in I^m. \forall D \in I^m. \\ (\Psi(C, D) \rightarrow (\chi(U, V) \wedge \Phi(X) \wedge \omega(U, V, X) \wedge \xi(U, V, X, C, D))). \end{aligned} \quad (9)$$

Recall that the formula $\Psi(C, D)$ expresses that the complex number $C + iD$ is contained in the set S . Intuitively speaking, the formula $\chi(U, V)$ will express that the variables U and V represent the real and imaginary parts of the eigenvalues $\lambda_1, \dots, \lambda_{m+b+s}$. The formula $\omega(U, V, X)$ will express that $(Q^{-1}X)_k = 0$ for $k = m + 1, \dots, m + b + s$. The formula $\xi(U, V, X, C, D)$ will express that $f(X, C + iD) \in K$.

More formally, let

$$\chi(U, V) = \bigwedge_{j=1}^{m+b+s} (h_j(U) = 0 \wedge a_j \leq U \leq b_j \wedge g_j(V) = 0 \wedge c_j \leq V \leq d_j).$$

Let

$$\omega(U, V, X) = \bigwedge_{k=m+1}^N \bigwedge_{s=0}^1 \left(\sum_{j=1}^n R_{s,k,j}(U_{r_j,k}, V_{r_j,k}) \cdot X_j = 0 \right),$$

Let $\xi(U, V, X, C, D)$ be the formula which is obtained from Φ by replacing each atom $P(X_1, \dots, X_n) \leq 0$ in Φ by the atom



$$P \left(\sum_{j=1}^n F_{1,j}(U, V)(C, D) \cdot X_j, \dots, \sum_{j=1}^n F_{n,j}(U, V)(C, D) \cdot X_j \right) \leq 0,$$

Note that this substitution can be performed in polynomial time. The polynomial P is given by a term t over the signature $\langle \mathbb{Z}, +, \times \rangle$. A term representing the new atom is obtained by substituting in the term t the occurrence of each variable X_k by the polynomial-size term $\sum_{j=1}^n F_{k,j}(U, V)(C, D) \cdot X_j$.



Now, observing that the formula $\chi(U, V)$ forces U and V to be equal respectively to the vector of real and imaginary parts of the eigenvalues $\lambda_1, \dots, \lambda_{m+b+s}$ it follows by construction that the $\mathbf{b}\text{-}\Sigma_{\leq}^{++}$ -sentence (9) is equivalent to the informal sentence (8) and hence expresses the existence of a trapped point. There is only one small argument required: By (7) the formula $\xi(\vec{\alpha}, \vec{\beta}, X, C, D)$ expresses that $\text{Re } f(X, C + iD) \in K$ rather than $f(X, C + iD) \in K$. But if $\Psi(C, D)$ holds true then $C + iD \in S$, so that $f(X, C + iD)$ is real-valued, for instance since it is contained in the closure of the orbit of $A^k x$ by the proof of Lemma 26.

Deciding the truth of the sentence (9) is therefore equivalent to deciding non-termination of the Escape Problem instance (A, K) .

The Pseudo-Skolem Problem is Decidable

Julian D’Costa  

Department of Computer Science,
University of Oxford, UK

Rupak Majumdar  

Max Planck Institute for Software Systems,
Kaiserslautern, Germany

Mahmoud Salamati  



Max Planck Institute for Software Systems,
Kaiserslautern, Germany

James Worrell  

Department of Computer Science,
University of Oxford, UK

Toghrul Karimov  

Max Planck Institute for Software Systems,
Saarland Informatics Campus,
Saarbrücken, Germany

Joël Ouaknine  

Max Planck Institute for Software Systems,
Saarland Informatics Campus,
Saarbrücken, Germany

Sadegh Soudjani  

Newcastle University, Newcastle upon Tyne, UK

Abstract

We study fundamental decision problems on linear dynamical systems in discrete time. We focus on *pseudo-orbits*, the collection of trajectories of the dynamical system for which there is an arbitrarily small perturbation at each step. Pseudo-orbits are generalizations of orbits in the topological theory of dynamical systems. We study the pseudo-orbit problem, whether a state belongs to the pseudo-orbit of another state, and the pseudo-Skolem problem, whether a hyperplane is reachable by an ϵ -pseudo-orbit for every ϵ . These problems are analogous to the well-studied orbit problem and Skolem problem on unperturbed dynamical systems. Our main results show that the pseudo-orbit problem is decidable in polynomial time and the Skolem problem on pseudo-orbits is decidable. The former extends the seminal result of Kannan and Lipton from orbits to pseudo-orbits. The latter is in contrast to the Skolem problem for linear dynamical systems, which remains open for proper orbits.

2012 ACM Subject Classification Theory of computation \rightarrow Design and analysis of algorithms

Keywords and phrases Pseudo-orbits, Orbit problem, Skolem problem, linear dynamical systems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.34

Funding *Julian D’Costa*: emmy.network foundation under the aegis of the Fondation de Luxembourg. *Rupak Majumdar*: DFG grant 389792660 as part of TRR 248 (see <https://perspicuous-computing.science>).

Joël Ouaknine: ERC grant AVS-ISS (648701), and DFG grant 389792660 as part of TRR 248 (see <https://perspicuous-computing.science>).

Joël Ouaknine is also affiliated with Keble College, Oxford as **emmy.network** Fellow.

James Worrell: EPSRC Fellowship EP/N008197/1.

1 Introduction

A (discrete-time) linear dynamical system in m dimensions is defined by a linear map $x \mapsto Ax$ for an $m \times m$ rational matrix A . The map specifies how an individual state (a real-valued vector in m dimensions) evolves over time; a trajectory starting from a state s is given by the sequence (s, As, A^2s, \dots) . Linear dynamical systems are fundamental models in many different domains of science and engineering, and the computability and complexity of decision problems for linear dynamical systems are of both theoretical and practical interest.



© Julian D’Costa, Toghrul Karimov, Rupak Majumdar, Joël Ouaknine, Mahmoud Salamati, Sadegh Soudjani, and James Worrell;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 34; pp. 34:1–34:21



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The *orbit* of a point s is the smallest set containing s and closed under the dynamic map. The *orbit problem* for linear dynamical systems asks, given s and t , if t is in the orbit of s [11]. In a seminal paper, Kannan and Lipton [12] showed that the orbit problem can be decided in polynomial time. However, a natural generalization of the orbit problem, the *Skolem problem*, in which we ask whether the orbit of a given state s intersects a given hyperplane, turns out to be notoriously difficult and remains open after many decades [20, 16]. A breakthrough occurred in the mid-1980s, when Mignotte *et al.* [14] and Vereshchagin [21] independently showed decidability in dimension 4 or less. These deep results make essential use of Baker’s theorem on linear forms in logarithms (which earned Baker the Fields Medal in 1970), as well as a p-adic analogue of Baker’s theorem due to van der Poorten. Unfortunately, little progress on that front has since been recorded.

The orbit and Skolem problems are defined on the exact dynamics of the linear system. In dynamical systems theory, one is often interested in “rough” dynamics of a system – in topological terms, we wish to study closed sets containing the orbit. Orbits arising from linear dynamics are usually not closed sets. Indeed the orbit of the point 1 under the map $x \mapsto \frac{1}{2}x$ does not contain the limit point 0. One way to retain closure is through *pseudo-orbits* [8], a concept going back several decades. A pseudo-orbit generalizes the orbit by allowing arbitrarily small imprecisions throughout the dynamics. For a precision $\epsilon > 0$, we say t is in the ϵ -pseudo-orbit of s if there is a sequence of points $(s = s_0, s_1, \dots, s_n = t)$ with $n > 0$ such that $\|As_i - s_{i+1}\| < \epsilon$ for each $i \in \{0, \dots, n - 1\}$. That is, an ϵ -pseudo-orbit contains a sequence of points that would be indistinguishable from an orbit if each state were known only up to precision ϵ . Finally, t is in the pseudo-orbit of s if it is in the ϵ -pseudo-orbit of s for all $\epsilon > 0$.

One can provide a computational analogue of pseudo-orbits (see [17]). Alice is simulating the trajectory of a dynamical system but in every iteration, her computation has a rounding error ϵ . An infinitely powerful adversary, Bob, rounds Alice’s result in an arbitrary fashion to a new state within a distance of ϵ of the actual outcome. A state t is pseudo-reachable from s iff Bob can fool Alice into believing that t is reachable in the simulation no matter how accurate her simulation is.

We can formulate analogous decision problems on pseudo-orbits. The *pseudo-orbit problem* asks, given a linear dynamical system and two states s and t , whether t is in the pseudo-orbit of s . The *hyperplane pseudo-reachability* (or pseudo-Skolem) problem asks, given a linear dynamical system, an initial state s , and a hyperplane, if there is an ϵ -pseudo orbit from s that intersects the hyperplane for every $\epsilon > 0$.

In this paper, we study decision problems for pseudo-orbits of linear dynamical systems. We show that **the pseudo-orbit problem is decidable in polynomial time** and that **the Skolem problem is decidable in full generality on pseudo-orbits**.

We proceed in two steps. First, we generalize Kannan and Lipton’s analysis to show that the pseudo-orbit problem can be decided in polynomial time. Our proof involves a careful examination of the eigenvalues of the matrix A , similar to Kannan and Lipton’s proof. More generally, we show that pseudo-reachability to a bounded semi-algebraic set is decidable.

Next, we consider the hyperplane pseudo-reachability (a.k.a. pseudo-Skolem) problem. Our proof again proceeds by a case analysis on the eigenvalues of A . The most challenging case is when there is an eigenvalue of modulus greater than 1. We analyze a series whose terms are polynomial-exponential functions of $n \in \mathbb{N}$ associated with the dynamics. We show that the infimum of this sum can be effectively computed. The proof of effective computability uses tools from Diophantine approximation as well as a reduction to the decision problem for the theory of real closed fields

We show that the dynamics pseudo-reaches the hyperplane in case the infimum of the above sum is 0. If the infimum is non-zero, we prove that we can find an effective bound N such that the dynamics pseudo-reaches the hyperplane iff, for sufficiently small ϵ , it pseudo-reaches the hyperplane within N steps.

Putting everything together, we conclude that the pseudo-Skolem problem is decidable.

Other related work. The study of pseudo-orbits goes back to Anosov, Bowen, and Conley [1, 3, 8]. Conley [8] formulated the fundamental theorem of dynamical systems: the iteration of any continuous, possibly non-linear, map on a compact metric space decomposes the space into a chain-recurrent part (the pseudo-orbit analogue of a period orbit) and a gradient-like part. Our results imply that deciding if a state is chain recurrent is decidable for linear systems.

In linear systems theory, *controllability* is a fundamental property of linear systems [19]. Controllability states that the system can be controlled from any point to any other point. However, this may require unboundedly large control actions. A pseudo-orbit can be seen as a stronger notion, where we ask if the dynamics can be controlled from a starting point to an ending point no matter how small the control input is: if a state belongs to the pseudo-orbit, then for every ϵ , there is a sequence of control inputs each bounded in norm by ϵ that steers the system to that state.

2 Linear Dynamical Systems

Notation. The sets of natural numbers (including zero), rational numbers, real numbers, and algebraic numbers are denoted by \mathbb{N} , \mathbb{Q} , \mathbb{R} , and $\overline{\mathbb{Q}}$, respectively. We assume a standard representation of algebraic numbers in terms of their defining polynomials, by which we can perform arithmetic operations and test equality in polynomial time in their representation (see, e.g., [6]).

For any column vector $x = [x_1, x_2, \dots, x_m]^\top \in \mathbb{R}^m$, we use the notations $\|x\|_2 := \sqrt{x^\top x}$ and $\|x\|_\infty := \max_i |x_i|$ to indicate respectively the two norm and infinity norm of x . For any matrix $A = [a_{ij}]_{i,j} \in \mathbb{R}^{m \times m}$, we define $\|A\|_2$ and $\|A\|_\infty$ to indicate respectively the (induced) two norm and infinity norm of A . Note that $\|Ax\|_2 \leq \|A\|_2 \|x\|_2$ and $\|Ax\|_\infty \leq \|A\|_\infty \|x\|_\infty$ for all $x \in \mathbb{R}^m$. We write $\mathbf{0} \in \mathbb{R}^m$ for the zero vector and $\mathbf{1} \in \mathbb{R}^m$ for the all-ones vector. We denote by $\rho(A)$ the spectral radius of a matrix A , which is the largest absolute value of the eigenvalues of A . For any $A \in \mathbb{R}^{m \times m}$ and any $\gamma > \rho(A)$, recall that there is a constant $c > 0$ such that $\|A^n\|_2 \leq c\gamma^n$ for all $n \in \mathbb{N}$.

Discrete-Time Linear Dynamical Systems. An m -dimensional discrete-time linear dynamical system is specified by an $m \times m$ matrix A of rational numbers. The *trajectory* determined by an initial state $x_0 \in \mathbb{R}^m$ is the sequence $(x_n)_{n \geq 0}$ given by

$$x_{n+1} = Ax_n, \quad (n \in \mathbb{N}).$$

We call the set $\mathcal{O}(A, x_0) := \{x_n \mid n \in \mathbb{N}\}$ the *orbit* of x_0 .

For any $\epsilon > 0$, an ϵ -perturbed linear dynamical system has state trajectories $(x_n)_{n \geq 0}$ such that

$$x_{n+1} = Ax_n + d_n, \quad (n \in \mathbb{N}),$$

where A is as before and $d_n \in [-\epsilon, \epsilon]^m$ for all n . For an initial state $x_0 \in \mathbb{R}^m$, we define the ϵ -pseudo-orbit $\tilde{\mathcal{O}}_\epsilon(A, x_0)$ of the dynamics as the set of states reachable in the perturbed dynamics. More formally, define

34:4 The Pseudo-Skolem Problem is Decidable

- for $n = 0$, $\tilde{\mathcal{O}}_\epsilon^{(n)}(A, x_0) := \{x_0\}$,
- for all $n \in \mathbb{N}$, $\tilde{\mathcal{O}}_\epsilon^{(n+1)}(A, x_0) := \{Ax + d \in \mathbb{R}^m \mid x \in \tilde{\mathcal{O}}_\epsilon^{(n)}(A, x_0), d \in [-\epsilon, \epsilon]^m\}$, and
- $\tilde{\mathcal{O}}_\epsilon(A, x_0) := \bigcup_{n \geq 0} \tilde{\mathcal{O}}_\epsilon^{(n)}(A, x_0)$.

Finally, we define the *pseudo-orbit* $\tilde{\mathcal{O}}(A, x_0) := \bigcap_{\epsilon > 0} \tilde{\mathcal{O}}_\epsilon(A, x_0)$ as the intersection of all the ϵ -pseudo orbits of x_0 , for all $\epsilon > 0$. Clearly, $\mathcal{O}(A, x) \subseteq \tilde{\mathcal{O}}(A, x)$ for any A and x .

We will make use of the following characterization, which follows directly from the definition: Any $t \in \tilde{\mathcal{O}}_\epsilon(A, s)$ is of the form $t = A^n s + \sum_{i=0}^{n-1} A^i d_{n-i-1}$ for some $n \in \mathbb{N}$ and some sequence of perturbations d_i with $\|d_i\|_\infty \leq \epsilon$.

We also need the following properties of $\tilde{\mathcal{O}}_\epsilon(A, x)$ and $\tilde{\mathcal{O}}(A, x)$.

▷ **Claim 1 (Transitivity).** For every A and $\epsilon > 0$, and for states $s, t, u \in \mathbb{R}^m$, if $t \in \tilde{\mathcal{O}}_\epsilon(A, s)$ and $u \in \tilde{\mathcal{O}}_\epsilon(A, t)$ then $u \in \tilde{\mathcal{O}}_\epsilon(A, s)$. If $t \in \tilde{\mathcal{O}}(A, s)$ and $u \in \tilde{\mathcal{O}}(A, t)$, then $u \in \tilde{\mathcal{O}}(A, s)$.

▷ **Claim 2 (Closure).** For every A , $\epsilon > 0$, and state $s \in \mathbb{R}^m$, the sets $\tilde{\mathcal{O}}_\epsilon(A, s)$ and $\tilde{\mathcal{O}}(A, s)$ are closed sets.

► **Problem 3 (Orbit problem).** Given $A \in \mathbb{Q}^{m \times m}$ and $s, t \in \mathbb{Q}^m$, decide whether $t \in \mathcal{O}(A, s)$.

A celebrated result of Kannan and Lipton [12] shows that the Orbit Problem is decidable in polynomial time.

► **Theorem 4 ([12]).** *The orbit problem is decidable in polynomial time.*

In this paper, we study the following problems.

► **Problem 5 (Pseudo-orbit problem).** Given $A \in \mathbb{Q}^{m \times m}$ and $s, t \in \mathbb{Q}^m$, decide whether $t \in \tilde{\mathcal{O}}(A, s)$.

► **Problem 6 (Hyperplane pseudo-reachability problem).** Given $A \in \mathbb{Q}^{m \times m}$, $s \in \mathbb{Q}^m$, and a hyperplane $c^T \cdot x = v$ for $c, v \in \mathbb{Q}^m$, decide whether $\tilde{\mathcal{O}}_\epsilon(A, s)$ intersects the hyperplane for all $\epsilon > 0$.

The following summarizes our main theorem.

► **Theorem 7 (Main Theorem).**

1. *The pseudo-orbit problem is decidable in polynomial time.*
2. *The hyperplane pseudo-reachability problem is decidable.*

The rest of the paper is devoted to the proof of this theorem.

2.1 Preliminaries

First we establish that pseudo-orbits can be translated with change of bases.

► **Proposition 8.** *For matrices $A, B, Q \in \mathbb{R}^{m \times m}$ with $A = QBQ^{-1}$ and for any $x \in \mathbb{R}^m$, we have $Q\tilde{\mathcal{O}}_{\gamma_2}(B, Q^{-1}x) \subseteq \tilde{\mathcal{O}}_\epsilon(A, x) \subseteq Q\tilde{\mathcal{O}}_{\gamma_1}(B, Q^{-1}x)$, where $\gamma_1 = \epsilon \|Q^{-1}\|_\infty$ and $\gamma_2 = \epsilon / \|Q\|_\infty$. Moreover, $\tilde{\mathcal{O}}(A, x) = Q\tilde{\mathcal{O}}(B, Q^{-1}x)$.*

We will use Proposition 8 with matrix A represented using the *Jordan canonical form*.

Jordan Decomposition. For a given rational square matrix A one can compute *change of basis matrix* Q and *Jordan normal form* J so that $A = QJQ^{-1}$ and $J = \text{diag}(J_1, J_2, \dots, J_z)$ with J_i representing the i^{th} Jordan block taking the following form

$$J_i = \begin{bmatrix} \Lambda_i & 1 & 0 & \dots & 0 & 0 \\ 0 & \Lambda_i & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \Lambda_i & 1 \\ 0 & 0 & 0 & \dots & 0 & \Lambda_i \end{bmatrix}, \quad (1)$$

where Λ_i denotes the i^{th} eigenvalue of A . The size of J_i is equal to the multiplicity of the eigenvalue Λ_i and is denoted by $\kappa(\Lambda_i)$.

Real Jordan form. For any $A \in \mathbb{R}^{n \times n}$ having complex eigenvalues, matrices Q and J in the Jordan normal form could have complex entries. In this case, the complex eigenvalues form complex conjugate pairs and give a *real Jordan form*: there are *real matrices* Q and J such that $A = QJQ^{-1}$ and $J = \text{diag}(J_1, J_2, \dots, J_z)$. The matrix J_i represents the i^{th} real Jordan block corresponding to either a real eigenvalue Λ_i or a complex pair $\Lambda_i = a_i \pm jb_i$. It is equal to (1) for real Λ_i and has the following form for the complex pair $\Lambda_i = a_i \pm jb_i$,

$$J_i = \begin{bmatrix} \Lambda_i & I_{2 \times 2} & 0_{2 \times 2} & \dots & 0_{2 \times 2} & 0_{2 \times 2} \\ 0_{2 \times 2} & \Lambda_i & I_{2 \times 2} & \dots & 0_{2 \times 2} & 0_{2 \times 2} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & \dots & \Lambda_i & I_{2 \times 2} \\ 0_{2 \times 2} & 0_{2 \times 2} & 0_{2 \times 2} & \dots & 0_{2 \times 2} & \Lambda_i \end{bmatrix}, \quad (2)$$

where with abuse of notation, we have indicated $\Lambda_i = \begin{bmatrix} a_i & -b_i \\ b_i & a_i \end{bmatrix}$. $I_{2 \times 2}$ and $0_{2 \times 2}$ denote identity and fully zero matrices of size 2 by 2.

The real Jordan normal form and the change of basis matrices Q and Q^{-1} can be computed in polynomial time (see [4] and also Appendix D).

Computing matrix powers. If $A = QJQ^{-1}$, then we have $A^n = QJ^nQ^{-1}$ for $n \in \mathbb{N}$, where $J^n = \text{diag}(J_1^n, J_2^n, \dots, J_z^n)$ and

$$J_i^n = \begin{bmatrix} \Lambda_i^n & n\Lambda_i^{n-1} & \binom{n}{2}\Lambda_i^{n-2} & \dots & \binom{n}{k-1}\Lambda_i^{n-k+1} \\ 0 & \Lambda_i^n & n\Lambda_i^{n-1} & \dots & \binom{n}{k-2}\Lambda_i^{n-k+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & n\Lambda_i^{n-1} \\ 0 & 0 & 0 & \dots & \Lambda_i^n \end{bmatrix}.$$

3 The pseudo-orbit problem is decidable in polynomial time

In this section, we show that Problem 5 is decidable in polynomial time. Fix a matrix A and let J be the real Jordan form for A . Proposition 8 shows that $\tilde{\mathcal{O}}(A, x)$ can be obtained from the pseudo-orbit $\tilde{\mathcal{O}}(J, x)$. Our decidability proof involves a case analysis on the modulus of the eigenvalues of J . We first consider the cases where J is a single block, i.e.,

$$J = \begin{bmatrix} \Lambda & I & & \\ & \Lambda & \ddots & \\ & & \ddots & I \\ & & & \Lambda \end{bmatrix} \text{ with } \Lambda = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \text{ and } I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ or } \Lambda = [r] \text{ and } I = [1], \quad (3)$$

with real matrix entries $a, b, r \in \mathbb{R}$.

We shall case split on the spectral radius $\rho(J)$, which is the absolute value of the unique eigenvalue of the Jordan block J . We consider three cases: $\rho(J) < 1$, $\rho(J) = 1$ and $\rho(J) > 1$. The following lemma will be useful in relating the first and third cases. Its proof is simply by reversing time.

► **Lemma 9** (Reversibility Lemma). *For any invertible matrix $A \in \mathbb{R}^{m \times m}$, $x \in \tilde{\mathcal{O}}_\epsilon(A, s)$ implies $s \in \tilde{\mathcal{O}}_\gamma(A^{-1}, x)$ with $\gamma = \epsilon \|A^{-1}\|_\infty$. Moreover,*

$$x \in \tilde{\mathcal{O}}(A, s) \iff s \in \tilde{\mathcal{O}}(A^{-1}, x). \quad (4)$$

► **Lemma 10** (Eigenvalues inside the unit circle). *Let $J \in \mathbb{R}^{m \times m}$ be a Jordan block of the form (3) with $\rho(J) < 1$. For every $s \in \mathbb{R}^m$,*

$$\tilde{\mathcal{O}}(J, s) = \mathcal{O}(J, s) \cup \{0\} = \overline{\mathcal{O}(J, s)},$$

where $\overline{\mathcal{O}(J, s)}$ denotes the closure of the orbit.

Proof. We prove the lemma by showing there is a constant $C > 0$ satisfying

$$\overline{\mathcal{O}(J, s)} \stackrel{*}{=} \mathcal{O}(J, s) \cup \{0\} \stackrel{**}{\subseteq} \tilde{\mathcal{O}}(J, s) \stackrel{\dagger}{\subseteq} \bigcap_{\epsilon > 0} \bigcup_{z \in \mathcal{O}(J, s)} \mathcal{B}(z, C\epsilon) \stackrel{\S}{\subseteq} \overline{\mathcal{O}(J, s)}, \quad (5)$$

where $\mathcal{B}(z, \epsilon) := \{y \in \mathbb{R}^m \mid \|z - y\|_2 \leq \epsilon\}$ is the closed ball with respect to two norm with center z and radius ϵ . It is easy to see that equality (*) holds since all the eigenvalues of J are inside the unit circle, $\lim_{n \rightarrow \infty} J^n = 0$, and 0 is the only limiting point of any state trajectory.

It is also easy to see that inclusion (**) is correct. Note that for any $\epsilon > 0$, $\mathcal{O}(J, s) \subseteq \tilde{\mathcal{O}}_\epsilon(J, s)$ and the set $\tilde{\mathcal{O}}_\epsilon(J, s)$ is closed by definition. Taking intersection over $\epsilon > 0$, we get $\mathcal{O}(J, s) \subseteq \tilde{\mathcal{O}}(J, s)$ with $\tilde{\mathcal{O}}(J, s)$ being a closed set. Therefore, $\overline{\mathcal{O}(J, s)} \subseteq \tilde{\mathcal{O}}(J, s)$.

We now choose a value of C which allows us to prove inclusion (†). First pick γ such that $\rho(J) < \gamma < 1$. Next choose c_1 to be a constant (which is guaranteed to exist) satisfying $\|J^n\|_2 \leq c_1 \gamma^n$ for all $n \in \mathbb{N}$, and finally set $C := c_1 m / (1 - \gamma)$. We show that $\tilde{\mathcal{O}}_\epsilon(J, s) \subseteq \bigcup_{z \in \mathcal{O}(J, s)} \mathcal{B}(z, C\epsilon)$ for any $\epsilon > 0$. Take any $x \in \tilde{\mathcal{O}}_\epsilon(J, s)$. Then there is a sequence (d_0, d_1, \dots) and $n \in \mathbb{N}$ such that $\|d_i\|_\infty \leq \epsilon$ and $x = J^n s + \sum_{i=0}^{n-1} J^i d_{n-i-1}$. Now

$$\|x - J^n s\|_2 = \left\| \sum_{i=0}^{n-1} J^i d_{n-i-1} \right\|_2 \leq \sum_{i=0}^{n-1} \|J^i\|_2 \|d_{n-i-1}\|_2 \leq \sum_{i=0}^{n-1} c_1 \gamma^i m \epsilon \leq \frac{c_1 m \epsilon}{1 - \gamma} = C\epsilon,$$

We then get $x \in \mathcal{B}(z, C\epsilon)$ for $z := J^n s \in \mathcal{O}(J, s)$.

The inclusion § can be proven by taking an arbitrary point $y \notin \overline{\mathcal{O}(J, s)}$ and showing that there is an $\epsilon > 0$ for which $y \notin \mathcal{B}(z, C\epsilon)$ for all $z \in \mathcal{O}(J, s)$. Note that the complement of $\overline{\mathcal{O}(J, s)}$ is an open set, which means there is a $\theta > 0$ such that $\mathcal{B}(y, \theta) \cap \overline{\mathcal{O}(J, s)} = \emptyset$. Taking ϵ such that $C\epsilon < \theta$ will give the intended result. ◀

Additionally, we prove the following lemma (that will be useful later) about the behaviour of pseudo-orbits when all eigenvalues are inside the unit circle.

► **Lemma 11.** *Let $A \in \mathbb{R}^{m \times m}$ and $s \in \mathbb{R}^m$. If $\rho(A) < 1$, then for every $\delta > 0$ there exists an effectively computable $N \in \mathbb{N}$ and $\epsilon > 0$ such that after time N , all ϵ -pseudo-orbits are contained inside the ball $\mathcal{B}(\mathbf{0}, \delta)$.*

Proof. Let $(x_n)_{n \in \mathbb{N}}$ denote an ϵ -pseudo-orbit starting from s with a sequence of disturbances $(d_n)_{n \in \mathbb{N}}$. Suppose $\rho(A) < 1$ and let $\gamma \in (\rho(A), 1)$. There is a constant $c > 0$ satisfying $\|A^n\|_2 \leq c\gamma^n$ for all n . Then we get

$$\begin{aligned} \|x_n\|_2 &= \left\| A^n s + \sum_{k=0}^{n-1} A^k d_{n-k-1} \right\|_2 \leq \|A^n\|_2 \|s\|_2 + \sum_{k=0}^{n-1} \|A^k\|_2 \|d_{n-k-1}\|_2 \\ &\leq c\gamma^n \|s\|_2 + \sum_{k=0}^{n-1} m\epsilon c\gamma^k \leq c\gamma^n \|s\|_2 + \frac{m\epsilon c}{1-\gamma}. \end{aligned}$$

Taking $\epsilon = \delta(1-\gamma)/(2mc)$ and N with $\gamma^N \|s\|_2 \leq \delta/(2c)$ gives the intended result. ◀

► **Lemma 12 (Eigenvalues outside the unit circle).** *Let $J \in \mathbb{R}^{m \times m}$ be a Jordan block of the form (3) with $\rho(J) > 1$. We have $\tilde{\mathcal{O}}(J, \mathbf{0}) = \mathbb{R}^m$ and $\tilde{\mathcal{O}}(J, s) = \mathcal{O}(J, s)$ if $s \neq \mathbf{0}$.*

Proof. In this case, J is invertible and all eigenvalues of J^{-1} are inside the unit circle. We apply the Reversibility Lemma 9 and Lemma 10.

$$x \in \tilde{\mathcal{O}}(J, s) \iff s \in \tilde{\mathcal{O}}(J^{-1}, x) \iff s \in \mathcal{O}(J^{-1}, x) \cup \{\mathbf{0}\} \iff s = \mathbf{0} \text{ or } x \in \mathcal{O}(J, s).$$

Therefore, any x is in $\tilde{\mathcal{O}}(J, s)$ if $s = \mathbf{0}$, and $\tilde{\mathcal{O}}(J, s) = \mathcal{O}(J, s)$ for $s \neq \mathbf{0}$. ◀

► **Lemma 13 (Eigenvalues on the unit circle).** *Let $J \in \mathbb{R}^{m \times m}$ be a Jordan block of the form (3) with $\rho(J) = 1$. For every $s \in \mathbb{R}^m$, we have $\tilde{\mathcal{O}}(J, s) = \mathbb{R}^m$.*

Proof. The key part of the proof is to show that $\mathbf{0} \in \tilde{\mathcal{O}}(A, s)$ for any s and for any A having the eigenvalues on the unit circle. Once we show this, we know that $s \in \tilde{\mathcal{O}}(A^{-1}, \mathbf{0})$ is true for any s and any matrix A due to the Reversibility lemma. Stated for the inverse of A and any x , we get $x \in \tilde{\mathcal{O}}(A, \mathbf{0})$. Since pseudo-orbits are transitive, we have $x \in \tilde{\mathcal{O}}(A, s)$ for any x and s , which is the intended result.

We show $\mathbf{0} \in \tilde{\mathcal{O}}(A, s)$ equivalently by replacing A with its Jordan form J and doing induction on the structure of J . The proof has two stages. The first stage is to show that $\mathbf{0} \in \tilde{\mathcal{O}}(J, s)$ for all s when J has a single block simple eigenvalues. The second stage is to show that we can sequentially increase the multiplicity of eigenvalues and multiple blocks.

Base case. Suppose $J = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$ with $a^2 + b^2 = 1$ or $J = r$ with $|r| = 1$. Observe that the multiplication by J does not increase the two norm of a vector. Hence setting

$$d_n = \begin{cases} -\epsilon \cdot \frac{Jx_n}{\|Jx_n\|_2} & \text{if } \|Jx_n\|_\infty > \epsilon, \\ -Jx_n & \text{otherwise,} \end{cases}$$

we obtain the ϵ -pseudo-orbit $(x_0 = s, x_1, x_2, \dots, x_m, \mathbf{0}, \mathbf{0}, \dots)$ from any s where $\|x_k\|_2 = \|x_{k-1}\|_2 - \epsilon$ for $k \leq m$, which gives $\mathbf{0} \in \tilde{\mathcal{O}}(J, s)$.

Inductive case. We show that if $\mathbf{0} \in \tilde{\mathcal{O}}(J_1, s_1)$ and $\mathbf{0} \in \tilde{\mathcal{O}}(J_2, s_2)$ for all s_1 and s_2 of appropriate dimensions, we also have $\mathbf{0} \in \tilde{\mathcal{O}}(J, s)$ with $J = \begin{bmatrix} J_1 & B \\ 0 & J_2 \end{bmatrix}$ for any B and any s with appropriate dimensions. Let us partition any state $x = (x^1, x^2)$ according to the dimensions of J_1 and J_2 . Let $\epsilon > 0$ and $s = (s^1, s^2)$. By the assumption, there exist ϵ -perturbations $(d_0^2, d_1^2, \dots, d_{N-1}^2)$ that bring s^2 to $\mathbf{0}$ under J_2 . Let $d_n = (\mathbf{0}, d_n^2)$ for $0 \leq n < N$ be a sequence of ϵ -perturbations for the linear system with mapping J . We obtain the sequence $(x_0 = s, x_1, \dots, x_N)$ with $x_N^2 = \mathbf{0}$: the ϵ -perturbations d_0, \dots, d_{N-1} have brought the second coordinate to $\mathbf{0}$. By the assumption, we also have $\mathbf{0} \in \tilde{\mathcal{O}}_\epsilon(J_1, x_N^1)$, which gives ϵ -perturbations (d_0^1, \dots, d_M^1) that bring x_N^1 to $\mathbf{0}$ under J_1 . Let us expand the sequence of perturbations for the linear system J with $d_{n+N} = (d_n^1, \mathbf{0})$ for $0 \leq n \leq M$. It is easy to see that (d_0, \dots, d_{N+M}) bring the system from s to $\mathbf{0}$ due to the structure of J that is upper triangular. ◀

We now consider the general case where J has multiple blocks.

► **Definition 14.** Let $J \in \mathbb{R}^{m \times m}$ be a real Jordan block matrix and $s \in \mathbb{R}^m$. We define

$$\Delta(J, s) := \begin{cases} \mathbb{R}^m & \text{if } \rho(J) = 1 \text{ or, } \rho(J) > 1 \text{ and } s = \mathbf{0}, \\ \{\mathbf{0}\} & \text{if } \rho(J) < 1, \\ \emptyset & \text{otherwise.} \end{cases}$$

The following lemma states that certain points in the pseudo-orbit of real Jordan blocks are ϵ -pseudo reachable exactly at any sufficiently large time step, for every $\epsilon > 0$. The lemma provides the flexibility to “synchronize” reaching parts of the state for different Jordan blocks.

► **Lemma 15 (Synchronization Lemma).** Let $J \in \mathbb{R}^{m \times m}$ be a Jordan block with eigenvalue λ . For $s \in \mathbb{R}^m$, $t \in \Delta(J, s)$ if and only if for every $\epsilon > 0$ there exists $N_\epsilon \in \mathbb{N}$ such that for all $N > N_\epsilon$, there exists an ϵ -pseudo-orbit $(x_i)_{i \in \mathbb{N}}$ of s under J such that $x_N = t$.

Proof.

- $|\lambda| < 1$ and $\Delta(J, s) = \{\mathbf{0}\}$. By Lemma 10, $\mathbf{0} \in \tilde{\mathcal{O}}(J, s)$ and hence for every $\epsilon > 0$, there exists N_ϵ such that $t = \mathbf{0}$ can be ϵ -pseudo reached at time N_ϵ . Now simply observe that once an ϵ -pseudo-orbit reaches $\mathbf{0}$, it can remain there forever by setting all future perturbations to zero. To prove the other direction, suppose $t \neq \mathbf{0}$. By Lemma 11, there must exist a time bound T such that for sufficiently small ϵ , all ϵ -pseudo-orbits of s after time T are contained in $\mathcal{B}(\mathbf{0}, \frac{\|t\|_2}{2})$. Hence for sufficiently small ϵ no N_ϵ with the the specified property can exist.
- $|\lambda| = 1$ and $\Delta(J, s) = \mathbb{R}^m$. In the proof of Lemma 13, for every $t \in \mathbb{R}^m$ and $\epsilon > 0$ we construct an ϵ -pseudo-orbit from s that visits $\mathbf{0}$ followed by t . Let N_ϵ be the number of steps required to ϵ -reach t . We can postpone visiting t to any time step $N > N_\epsilon$ by simply waiting at the point $\mathbf{0}$ for $N - N_\epsilon$ steps.
- $|\lambda| > 1$, $s = \mathbf{0}$ and $\Delta(J, s) = \mathbb{R}^m$. Similarly to the case above, in Lemma 12 for each ϵ we construct an ϵ -pseudo-orbit that visits t at time N_ϵ , and reaching t can be delayed arbitrarily by spending a necessary number of steps at $\mathbf{0}$ at the beginning.
- $|\lambda| > 1$, $s \neq \mathbf{0}$ and $\Delta(J, s) = \emptyset$. Let $t \in \mathbb{R}^m$. In this case, observe that there must exist a time bound T such that for sufficiently small ϵ , all ϵ -pseudo-orbits of s after time T are contained outside $\mathcal{B}(\mathbf{0}, 2\|t\|_2)$. Hence for sufficiently small ϵ no N_ϵ with the the specified property can exist. ◀

There are two modes of pseudo reachability: via orbit, or at larger and larger time steps for smaller ϵ .

► **Lemma 16.** *Let $A \in \mathbb{R}^{m \times m}$ and $s, t \in \mathbb{R}^m$. If there exists N such that for every ϵ , t is ϵ -pseudo-reachable from s within the first N steps, then $t \in \mathcal{O}(A, s)$.*

Proof. Suppose such N exists. By continuity of the map $x \mapsto Ax$, for every $\delta > 0$ there exists $\epsilon > 0$ such that for every $\epsilon' < \epsilon$ and ϵ' -pseudo-orbit $(x_i)_{i \in \mathbb{N}}$, $\|x_i - A^i s\|_2 < \delta$ for $0 \leq i < N$. Hence the intersection of the first N elements of all ϵ -pseudo-orbits is exactly $\{s, As, \dots, A^{N-1}s\}$. ◀

► **Lemma 17.** *For $J = \text{diag}(J_1, \dots, J_l)$ in real Jordan normal form and $s \in \mathbb{R}^m$,*

$$\tilde{\mathcal{O}}(J, s) = \mathcal{O}(J, s) \cup \prod_{i=1}^l \Delta(J_i, s_i).$$

Proof. Suppose $t = (t_1, \dots, t_l) \in \prod_{i=1}^l \Delta(J_i, s_i)$. That is, for every ϵ and $1 \leq i \leq l$ there exists an ϵ -pseudo-orbit $(x_j^i)_{j \in \mathbb{N}}$ of s_i under J_i that reaches t_i . By Lemma 15, for every ϵ there exist ϵ -pseudo-orbits $(y_j^i)_{j \in \mathbb{N}}$ of s_1, \dots, s_l that reach t_1, \dots, t_l , respectively, at the same time N . That is, $y_N^i = t_i$ for $1 \leq i \leq m$. Hence $(y_i^1, \dots, y_i^l)_{i \in \mathbb{N}}$ is an ϵ -pseudo-orbit of s under J that reaches t .

Now suppose $t \in \tilde{\mathcal{O}}(J, s) \setminus \mathcal{O}(J, s)$. We prove, by a case analysis on J_i , that $t_i \in \Delta(J_i, s_i)$ for $1 \leq i \leq l$. The main idea is that if t is pseudo-reachable but not reachable, then in order to reach it via an ϵ -pseudo-orbit one will need longer and longer time horizons as $\epsilon \rightarrow 0$ (Lemma 16).

1. $\rho(J_i) < 1$. Since t is not in the orbit, we can find a sequence $N_1 < N_2 < \dots$ of time steps and $\epsilon_1 > \epsilon_2 > \dots$ of perturbations such that t is ϵ_j -reachable from s earliest at time N_j . In particular, t_i is ϵ_j reachable from s_i at time N_j for every j . But by Lemma 11 this means that $|t_i| < \delta$ for every $\delta > 0$. Hence $t_i = \mathbf{0} \in \Delta(J_i, s_i)$.
2. $\rho(J_i) = 1$. Since in this case $\Delta(J_i, s_i) = \mathbb{R}^{\kappa(i)}$, trivially $t_i \in \Delta(J_i, s_i)$.
3. $\rho(J_i) > 1$ and $s_i = \mathbf{0}$. Since in this case too $\Delta(J_i, s_i) = \mathbb{R}^{\kappa(i)}$, trivially $t_i \in \Delta(J_i, s_i)$.
4. $\rho(J_i) > 1$ and $s_i \neq \mathbf{0}$. This case cannot arise, as similarly to Case 1, one can argue that if pseudo-reaching t_i requires larger and larger time steps as $\epsilon \rightarrow 0$, then $|t_i| > \delta$ for every δ . But in this case no such t_i can exist. ◀

Proof. (of Theorem 7(1)). We now put everything together to show the pseudo-orbit problem is decidable in polynomial time. Given $A \in \mathbb{Q}^{m \times m}$, and $s, t \in \mathbb{Q}^m$, we compute (in polynomial time) matrices $Q, J, Q^{-1} \in (\mathbb{R} \cap \overline{\mathbb{Q}})^{m \times m}$ such that $A = QJQ^{-1}$ and J is in real Jordan normal form (Appendix D). Then, we compute $t' = Q^{-1}t$ and $s' = Q^{-1}s$, and by Proposition 8 we have that $t \in \tilde{\mathcal{O}}(A, s)$ if and only if $t' \in \tilde{\mathcal{O}}(J, s')$. It remains to decide whether $t' \in \tilde{\mathcal{O}}(J, s')$. For this we use the characterization described in Lemma 17. To decide whether $t' \in \mathcal{O}(J, s')$, observe that $Q^{-1}t \in \mathcal{O}(J, Q^{-1}s) \iff t \in \mathcal{O}(A, s)$, and whether $t \in \mathcal{O}(A, s)$ is an instance of the Orbit Problem and can be decided in polynomial time.¹ Finally, it remains to check whether $t_i \in \Delta(J_i, s_i)$ for each block J_i , which can be done easily given the simplicity of $\Delta(J_i, s_i)$. ◀

We end the section with an application of Theorem 7(1). A set S is *pseudo-reachable* from s under A if for every $\epsilon > 0$, there exists a point $x_\epsilon \in S$ that is ϵ -pseudo-reachable from s under A . An *algebraic* set is the set of zeros of a collection of polynomials. A *semialgebraic*

¹ Technically, [12] consider the orbit problem for rational inputs and we require the orbit problem where the input can contain algebraic numbers. However, a polynomial time algorithm is still possible.

34:10 The Pseudo-Skolem Problem is Decidable

set is a union of algebraic sets and projections of algebraic sets. We show (in Appendix B.2) that we can decide if a bounded semialgebraic set is pseudo-reachable, by reducing the problem to the pseudo-orbit problem.

► **Theorem 18.** *Given $A \in \mathbb{Q}^{m \times m}$, $x_0 \in \mathbb{Q}^m$, and a bounded semialgebraic set S , it is decidable if S is pseudo-reachable from x_0 under A .*

4 Hyperplane pseudo-reachability is decidable

In this section, we prove Theorem 7(2). First we consider the case where we are given:

- a hyperplane $H = \{x \in \mathbb{R}^m : c^\top x = v\}$ with $(c, v) \in (\mathbb{R} \cap \overline{\mathbb{Q}})^m \times (\mathbb{R} \cap \overline{\mathbb{Q}})$,
- $J = \text{diag}(J_1, \dots, J_z) \in (\mathbb{R} \cap \overline{\mathbb{Q}})^{m \times m}$ in real Jordan normal form, and
- a starting point $x_0 \in (\mathbb{R} \cap \overline{\mathbb{Q}})^m$.

We show how to decide if for every $\epsilon > 0$ there exists an ϵ -pseudo-orbit $(x_i)_{i \in \mathbb{N}}$ of x_0 under J that *hits* the hyperplane H , i.e. $c^\top x_N - v = 0$ for some $N \in \mathbb{N}$.

A block J_i is *relevant* with respect to hyperplane $H = \{x : c^\top x = v\}$ if the coefficients of c at the coordinates corresponding to J_i are not all 0. Intuitively, dimensions corresponding to blocks that are not relevant can simply be omitted from the analysis as they do not play a role in determining whether a point is in H or not. *Relevant eigenvalues* of J are the eigenvalues of relevant blocks. The *relevant spectral radius*, written $\rho_H(J)$, is the largest modulus of all relevant eigenvalues. Our proof is based on a case analysis on the relevant spectral radius of J . We shall see that the proof is simple when the relevant spectral radius is ≤ 1 but requires more technical ideas when it is > 1 .

► **Lemma 19** (Case $\rho_H(J) \leq 1$). *Fix a matrix J in real Jordan normal form, a starting state x_0 , and a hyperplane $H = \{x : c^\top x = v\}$.*

1. *If $\rho_H(J) = 1$, then H is pseudo-reachable.*
2. *If $\rho_H(J) < 1$ and $\mathbf{0} \in H$ then H is pseudo-reachable. If $\rho_H(J) < 1$ and $\mathbf{0} \notin H$, there exists an effectively computable time bound N such that H is pseudo-reachable if and only if there exists $0 \leq i \leq N$ such that $J^i x_0 \in H$ (that is, H is reachable from x_0 under J after at most N steps).*

Proof. First suppose $\rho_H(J) = 1$. We write $J = \text{diag}(J_h, J_r)$, where $\rho_H(J_h) = 1$ and $\rho_H(J_r) < 1$ (observe that wlog we can assume the blocks of J have non-decreasing spectral radius when listed from top to bottom) and correspondingly set $s = (s_h, s_r)$, $c = (c_h, c_r)$. Note that $c_h \neq 0$ by the relevance of at least one of eigenvalues of modulus 1.

By Lemma 10 we know $\mathbf{0} \in \tilde{\mathcal{O}}(J_r, s_r)$. By Lemma 13, we can select y such that $c_h^\top y - v = 0$ and $y \in \tilde{\mathcal{O}}(J_h, s_h)$. Therefore, invoking Lemma 15, for every $\epsilon > 0$ we can find $N \in \mathbb{N}$ and construct ϵ -pseudo-orbits $(x_n^h)_{n \in \mathbb{N}}$ and $(x_n^r)_{n \in \mathbb{N}}$ such that $x_N^h = y$ and $x_N^r = \mathbf{0}$, which implies that for the ϵ -pseudo-orbit $x_n = (x_n^h, x_n^r)$, $c^\top x_N - v = c_h^\top y + c_r^\top \mathbf{0} - v = 0$ as desired.

Now suppose $\rho_H(J) < 1$.

Case 1: $v = 0$. Since $\mathbf{0} \in H$ and the origin is pseudo-reachable from x_0 (Lemma 10), H is pseudo-reachable.

Case 2: $v \neq 0$. Using Lemma 11, and setting $\delta = |v|/(2\|c\|_2)$, we can find $\epsilon > 0$ and horizon $N \in \mathbb{N}$ after which every ϵ -pseudo-orbit is trapped in $\mathcal{B}(0, \delta)$. Thus, the hyperplane cannot be pseudo-reached after time N , as the hyperplane does not intersect with $\mathcal{B}(0, \delta)$. It remains to check if the hyperplane is pseudo-reachable at any of the first N time-steps. In fact, for a bounded time interval, a hyperplane is pseudo-reachable iff it is reachable. This is

because the effect of finitely many disturbance terms (d_0, \dots, d_{N-1}) can be made arbitrarily small for small enough ϵ . Therefore, decidability in this case only requires checking if the bounded orbit $(y_n)_{0 \leq n \leq N}$ hits the hyperplane before the time horizon N , that is, if there exists a time-step $0 \leq n \leq N$ such that $c^\top y_n - v = 0$, which is clearly decidable. ◀

We now consider the case $\rho_H(J) > 1$. The main ideas of our proof are as follows:

1. A point x_n in the ϵ -pseudo orbit belongs to the hyperplane (c, v) if $c^\top x_n - v = 0$. In particular, $c^\top x_n - v$ can be written as a sum over exponential polynomials in eigenvalues of different sizes.
2. We factor out the scaling factor corresponding to the top eigenvalues, leaving a sum over normalized eigenvalues, together with a sum over disturbances (of order ϵ) and additional terms which go to zero with large n .
3. We relate hyperplane pseudo-reachability to the limit inferior of the sum over normalized eigenvalues. If the limit is zero, we show the hyperplane is pseudo-reachable. If the limit is positive, we show there is an effective bound N such that if the hyperplane is pseudo-reachable, it is reachable within N steps.
4. We apply results from Diophantine approximation and the theory of reals to compute the limit inferior of the sum over normalized eigenvalues.

Fix $J = \text{diag}(J_1, \dots, J_l) \in (\mathbb{R} \cap \overline{\mathbb{Q}})^{m \times m}$, a starting point $x_0 \in (\mathbb{R} \cap \overline{\mathbb{Q}})^m$, and a hyperplane $H = \{x \in \mathbb{R}^m \mid c^\top x = v\}$ with $c, v \in (\mathbb{R} \cap \overline{\mathbb{Q}})^m$. We assume without loss of generality that all blocks are relevant.

Step 1: Analysing $c^\top x_n - v$. Let $L = \rho_H(J) > 1$ be the largest modulus of a relevant eigenvalue of J and suppose the blocks are arranged in non-increasing order of the modulus of eigenvalues. In particular, let $t \leq l$ be such that the first t blocks (t for “top”) have $\rho(J_1) = \dots = \rho(J_t) = L > 1$. We call the eigenvalues of these blocks the *top eigenvalues*. The remaining blocks satisfy $L > \rho(J_{t+1}) \geq \dots \geq \rho(J_l)$.

Let $(d_i)_{i \in \mathbb{N}}$ be a sequence of perturbations and $(x_i)_{i \in \mathbb{N}}$ the resulting pseudo-orbit. We have that for all time steps n ,

$$c^\top x_n - v = c^\top \left(J^n x_0 + \sum_{k=0}^{n-1} J^k d_{n-k-1} \right) - v = \sum_{i=1}^l \left(c^i J_i^n x_0^i + c^i \sum_{k=0}^{n-1} J_i^k d_{n-k-1}^i \right) - v,$$

where for all $1 \leq i \leq l$, c^i , x_n^i , d_n^i are projections of c^\top , x_n and d_n , respectively, onto the coordinates governed by J_i . Observe that c^i is a row vector for every i .

Step 2: Normalized sum. We define a normalized version of this sum by factoring out L^n (the size of the top eigenvalues) and n^D , where we define D in such a way that we normalize polynomials in n that appear in the sum. Observe that for $1 \leq i \leq t$ (the top eigenvalues),

$$c^i J_i^n = \begin{cases} \left[p_1^i(n) \lambda^n + \overline{p_1^i(n) \lambda^n} \quad \dots \quad p_{2\kappa(i)}^i(n) \lambda^n + \overline{p_{2\kappa(i)}^i(n) \lambda^n} \right] & \text{if } J_i \text{ has eigenvalues } \lambda, \bar{\lambda} \\ \left[p_1^i(n) \rho^n \quad \dots \quad p_{\kappa(i)}^i(n) \rho^n \right] & \text{if } J_i \text{ has a single eigenvalue } \rho \end{cases}$$

for polynomials $p_1^i, \dots, p_{\kappa(i)}^i$ (with algebraic coefficients) where $\kappa(i)$ is the multiplicity of the block J_i .

We define D to be the largest number such that the monomial n^D appears with a non-zero coefficient in at least one of $c^i J_i^n$ for $1 \leq i \leq t$. (Note that if all entries of c are non-zero $D + 1$ is equal to the largest multiplicity of a top eigenvalue block of J , as can be seen from the description of powers of a Jordan block in Section 2.)

34:12 The Pseudo-Skolem Problem is Decidable

We can now define

$$f(n) := \frac{c^\top \cdot x_n - v}{L^n n^D} = \sum_{i=1}^l \left(c^i \frac{J_i^n}{L^n n^D} x_0^i + c^i \sum_{k=0}^{n-1} \frac{J_i^k}{L^n n^D} d_{n-k-1}^i \right) - \frac{v}{L^n n^D}$$

For notational convenience we define vector-valued functions $g^i(n) := c^i \frac{J_i^n}{L^n n^D}$ for $1 \leq i \leq l$. The following technical lemma summarizes the relevant properties of these scaled terms.

► **Lemma 20 (Normalization Lemma).**

1. For $1 \leq i \leq t$ (top eigenvalues), $\|g^i(n)\|_\infty = O(1)$ (with respect to n).
2. For $t+1 \leq i \leq l$ (non-top eigenvalues), $\lim_{n \rightarrow \infty} \|g^i(n)\|_\infty = 0$.
3. There exists $1 \leq j \leq t$ and effectively computable $N \in \mathbb{N}$ and $C > 0$ such that $n > N \implies \|g^j(n)\|_\infty > C$.

Proof. We address each point individually.

- 1: For $1 \leq i \leq t$ let J_i have eigenvalues λ and $\bar{\lambda}$ (the case where J_i has a single real eigenvalue is similar but simpler) and observe that

$$g^i(n) = \left[\frac{p_1^i(n)}{n^D} \left(\frac{\lambda}{L}\right)^n + \frac{\overline{p_1^i(n)}}{n^D} \left(\frac{\bar{\lambda}}{L}\right)^n \quad \dots \quad \frac{p_{2\kappa(i)}^i(n)}{n^D} \left(\frac{\lambda}{L}\right)^n + \frac{\overline{p_{2\kappa(i)}^i(n)}}{n^D} \left(\frac{\bar{\lambda}}{L}\right)^n \right].$$

By the definition of top eigenvalues, $|\lambda| = L$ and thus $\frac{\lambda}{L}$ and $\frac{\bar{\lambda}}{L}$ have modulus 1. By construction of n^D , the polynomials $p_1^i(n), \dots, p_{2\kappa(i)}^i(n)$ all have degree at most D and hence the terms $\frac{p_1^i(n)}{n^D}, \dots, \frac{p_{2\kappa(i)}^i(n)}{n^D}$ are bounded from above by a constant.

- 2: For $t+1 \leq i \leq l$ let J_i have eigenvalues λ and $\bar{\lambda}$ and observe that

$$g^i(n) = \left[\frac{p_1^i(n)}{n^D} \left(\frac{\lambda}{L}\right)^n + \frac{\overline{p_1^i(n)}}{n^D} \left(\frac{\bar{\lambda}}{L}\right)^n \quad \dots \quad \frac{p_{\kappa(i)}^i(n)}{n^D} \left(\frac{\lambda}{L}\right)^n + \frac{\overline{p_{\kappa(i)}^i(n)}}{n^D} \left(\frac{\bar{\lambda}}{L}\right)^n \right].$$

By construction $|\lambda| < L$ and thus $\gamma := \frac{\lambda}{L}$ and $\bar{\gamma}$ have moduli $|\gamma|, |\bar{\gamma}| < 1$. The polynomials $p_1^i(n), \dots, p_{\kappa(i)}^i(n)$ may not be asymptotically bounded by n^D (since n^D was constructed only considering top eigenvalues). However, it is clear that the exponentially vanishing $\left(\frac{\lambda}{L}\right)^n$ and $\left(\frac{\bar{\lambda}}{L}\right)^n$ will dominate the polynomials and all entries of $g^i(n)$ will thus vanish.

- 3: Observe that by construction of n^D , there must exist a top eigenvalue block J_j ($1 \leq j \leq t$) for which at least one polynomial in $c^j J_j^n$ has degree D . Let $r > D$ be the multiplicity of the block J_j , which has the form of a real Jordan matrix with a single block (Eq. (3)) with sub-blocks Λ . One can write

$$c^j J_j^n = \begin{bmatrix} c_r^j & c_{r-1}^j & \dots & c_0^j \end{bmatrix} \begin{pmatrix} \Lambda^n & n\Lambda^{n-1} & \binom{n}{2}\Lambda^{n-1} & \dots & \binom{n}{r-1}\Lambda^{n-r+1} \\ 0 & \Lambda^n & n\Lambda^{n-1} & \dots & \binom{n}{r-2}\Lambda^{n-r+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & n\Lambda^{n-1} \\ 0 & 0 & 0 & \dots & \Lambda^n \end{pmatrix}, \quad (6)$$

where c_k^j for $1 \leq k \leq r$ corresponds to a row vector of size two or one, respectively, when Λ is a 2×2 or 1×1 matrix. Analyzing this product, we see that $c_r^j, \dots, c_{D+1}^j = \mathbf{0}$, $c_D^j \neq \mathbf{0}$ and the single entry of $c^j J_j^n$ whose polynomial component has degree D is exactly $c_D^j \binom{n}{D} \Lambda^{n-D}$.

We define $\hat{\Lambda} := \Lambda/L$. Note that $\|\hat{\Lambda}\|_2 = 1$. Now observe that for this block J_j , we have

$$g^j(n) = \frac{1}{L^{n_n D}} c^j J_i^n = c_D^j \frac{1}{D!} \hat{\Lambda}^n + \frac{1}{n} (O(1)).$$

Therefore, there exists sufficiently large N such that for all $n \in \mathbb{N}$,

$$n > N \implies \left\| \frac{1}{L^{n_n D}} c^j J_j^n \right\|_\infty > \frac{1}{2} \left\| c_D^j \frac{1}{D!} \hat{\Lambda}^n \right\|_\infty > \frac{\|c_D^j\|_2}{4D!}.$$

Thus we have shown Point 3 with $C = \frac{\|c_D^j\|_2}{4D!}$. \blacktriangleleft

Step 3: Conditions for reachability and non-reachability. Now we are ready to attack our original problem. Going back, H is ϵ -pseudo-reachable if and only if $f(n) = 0$ for some disturbance sequence $(d_i)_{i \in \mathbb{N}}$ with $d_i \in [-\epsilon, \epsilon]^m$ for all i . We analyze how $f(n)$ can be brought to 0 in this way.

► **Lemma 21.** *Let*

$$D = \liminf_{n \rightarrow \infty} \left| \sum_{i=1}^t g^i(n) x_0^i \right|. \quad (7)$$

If $D = 0$, then H is pseudo-reachable. If $D > 0$, there exists a computable time bound N such that H is pseudo-reachable if and only if it is reachable (in the standard sense) within the first N steps.

Proof. Suppose $D = 0$. Take an arbitrary $\epsilon > 0$. We argue that H is ϵ -pseudo-reachable. Recall that

$$\begin{aligned} f(n) &= \sum_{i=1}^l \left(c^i \frac{J_i^n}{L^{n_n D}} x_0^i + c^i \sum_{k=0}^{n-1} \frac{J_i^k}{L^{n_n D}} d_{n-k-1}^i \right) - \frac{v}{L^{n_n D}} \\ &= \sum_{i=1}^l \left(g^i(n) x_0^i + c^i \sum_{k=0}^{n-1} \frac{J_i^k}{L^{n_n D}} d_{n-k-1}^i \right) - \frac{v}{L^{n_n D}}. \end{aligned}$$

Let I be such that $\|g^I(n)\|_\infty > C$, for $C > 0$ and sufficiently large n (Point 3 of the Normalization Lemma). We construct a pseudo-orbit with all perturbations set to zero except d_0^I and obtain

$$f(n) = c^I \frac{J_I^{n-1}}{L^{n_n D}} d_0^I + \sum_{i=1}^t g^i(n) x_0^i + \sum_{i=t+1}^l c^i \frac{J_i^n}{L^{n_n D}} x_0^i - \frac{v}{L^{n_n D}}.$$

Intuitively, we will use the term $c^I \frac{J_I^{n-1}}{L^{n_n D}} d_0^I$ to cancel out the remaining summands above, but we have to argue that this can be done using a disturbance of size at most ϵ . Moreover, observe that $c^I \frac{J_I^{n-1}}{L^{n_n D}}$ is very close to $g^I(n)$. Formally, we first find N large enough such that

- $\|g^I(N)\|_\infty > C$,
- $\left\| \sum_{i=t+1}^l c^i \frac{J_i^N}{L^{N N D}} x_0^i - \frac{v}{L^{N N D}} \right\|_\infty < \frac{C^2}{\|J_i\|_\infty} \frac{\epsilon}{2}$ (possible because for $t+1 \leq i \leq l$, $\rho(J_i) < 1$ and $L > 1$), and
- $\left\| \sum_{i=1}^t g^i(N) x_0^i \right\|_\infty < \frac{C^2}{\|J_i\|_\infty} \frac{\epsilon}{2}$ (possible because $\liminf_{n \rightarrow \infty} \left\| \sum_{i=1}^t g^i(n) x_0^i \right\| = 0$).

34:14 The Pseudo-Skolem Problem is Decidable

Finally, we determine the value of d_0^I . Without loss of generality, assume that $g^I(N)$ is of the form $[C' \ \dots]$ where $|C'| > C$, that is the first entry of $g^I(N)$ is large. We then observe that $c^I \frac{J_I^{N-1}}{L^N N^D} d_0^I = g^I(N) J_I^{-1} d_0^I$ and set

$$d_0^I = J_I \cdot \left[-\frac{1}{C'} \left(\sum_{i=1}^t g^i(N) x_0^i + \sum_{i=t+1}^l c^i \frac{J_i^N}{L^N N^D} x_0^i - \frac{v}{L^N N^D} \right) \ 0 \ 0 \ \dots \ 0 \right]^\top$$

to obtain

$$c^I \frac{J_I^{N-1}}{L^N N^D} d_0^I = - \left(\sum_{i=1}^t g^i(N) x_0^i + \sum_{i=t+1}^l c^i \frac{J_i^N}{L^N N^D} x_0^i - \frac{v}{L^N N^D} \right)$$

and hence $f(N) = 0$.

Now suppose $D > 0$. Recall

$$\begin{aligned} f(n) &= \sum_{i=1}^l \left(g^i(n) x_0^i + c^i \sum_{k=0}^{n-1} \frac{J_i^k}{L^n n^D} d_{n-k-1}^i \right) - \frac{v}{L^n n^D} \\ &= \sum_{i=1}^t g^i(n) x_0^i + \sum_{i=t+1}^l c^i \frac{J_i^n}{L^n n^D} x_0^i + \sum_{i=1}^l c^i \sum_{k=0}^{n-1} \frac{J_i^k}{L^n n^D} d_{n-k-1}^i - \frac{v}{L^n n^D}. \end{aligned}$$

In this case we shall construct a time bound N after which for all sufficiently small value of ϵ , the term $\sum_{i=1}^t g^i(n) x_0^i$ will dominate the other summands. Let $2\Delta > 0$ be a lower bound on $\liminf_{n \rightarrow \infty} \left| \sum_{i=1}^t g^i(n) x_0^i \right| > 0$. We shall see how to obtain such a bound effectively later (Lemma 22). We compute N with the following properties.

- For all $n > N$, $\left| \sum_{i=1}^t g^i(n) x_0^i \right| > \Delta$. Possible because $\liminf_{n \rightarrow \infty} \left| \sum_{i=1}^t g^i(n) x_0^i \right| > 2\Delta$.
- For all $n > N$, $\left| \sum_{i=t+1}^l c^i \frac{J_i^n}{L^n n^D} x_0^i \right|, \left| \frac{v}{L^n n^D} \right| \ll \Delta$. The former is possible because for $t+1 \leq i \leq l$, $\rho(J_i) < L$.
- For sufficiently small ϵ , for all $n > N$, $\left| c^i \sum_{k=0}^{n-1} \frac{J_i^k}{L^n n^D} d_{n-k-1}^i \right| \ll \Delta$ for $1 \leq i \leq l$. To see that this is always possible, observe that

$$\left| c^i \sum_{k=0}^{n-1} \frac{J_i^k}{L^n n^D} d_{n-k-1}^i \right| \leq \sum_{k=0}^{n-1} \left\| c^i \frac{J_i^k}{L^n n^D} \right\|_\infty M \epsilon \text{ (where fixed } M \text{ bounds the matrix dimension)}$$

and

$$\lim_{n \rightarrow \infty} \sum_{k=0}^n \left\| c^i \frac{J_i^k}{L^n n^D} \right\|_\infty \leq \lim_{n \rightarrow \infty} \sum_{k=0}^n \left\| c^i \frac{1}{L^{n-k}} \frac{J_i^k}{L^k k^D} \right\|_\infty = \lim_{n \rightarrow \infty} \sum_{k=0}^n \left\| \frac{1}{L^{n-k}} g^i(k) \right\|_\infty.$$

Recalling Point 1 of the Normalization Lemma, $\|g^i(n)\|_\infty = O(1)$ and hence

$$\lim_{n \rightarrow \infty} \sum_{k=0}^n \left\| \frac{1}{L^{n-k}} g^i(k) \right\|_\infty = O(1),$$

by bounding the sum $\sum_{k=0}^n \left\| \frac{1}{L^{n-k}} g^i(k) \right\|_\infty$ from above by a geometric sequence. Therefore, $\sum_{k=0}^{n-1} \left\| c^i \frac{J_i^k}{L^n n^D} \right\|_\infty M \epsilon$ can be made $\ll \Delta$ by choosing ϵ to be sufficiently small.

Once we have chosen N , by the properties above we will have that for all $n > N$, for sufficiently small ϵ ,

$$|f(n)| \geq \left| \sum_{i=1}^t g^i(n) x_0^i \right| - \left| \sum_{i=t+1}^l c^i \frac{J_i^n}{L^n n^D} x_0^i + \sum_{i=1}^l c^i \sum_{k=0}^{n-1} \frac{J_i^k}{L^n n^D} d_{n-k-1}^i - \frac{v}{L^n n^D} \right| > 0.$$

Therefore, H is pseudo-reachable if and only if for every $\epsilon > 0$, H is ϵ -pseudo-reachable within the first N steps. By Lemma 16, this is the case if and only if H is reachable within the first N steps. ◀

Step 4: Analyzing $\liminf_{n \rightarrow \infty} |\sum_{i=1}^t g^i(n)x_0^i|$. Consider a single term $g^i(n)x_0^i$. Writing $x_0^i = [X_0 \ X_1 \ \dots \ X_z]^\top$, where $X_1, \dots, X_z \in \mathbb{R}$, we have

$$g^i(n)x_0^i = \sum_{r=1}^z \left(\frac{p_r^i(n)}{n^D} \left(\frac{\lambda}{L} \right)^n + \overline{\frac{p_r^i(n)}{n^D} \left(\frac{\lambda}{L} \right)^n} \right) X_z.$$

Let $\gamma_i = \frac{\lambda}{L}$. Note that $|\gamma_i| = 1$. By the construction of n^D , none of the polynomials have a term of degree higher than D . Therefore, we can absorb the constants X_r and the monomial n^D into the polynomials, sum the terms up, and write them as polynomials in $\frac{1}{n}$. That is,

$$g^i(n)x_0^i = q^i(1/n)\gamma_i^n + \overline{q^i(1/n)\gamma_i^n}$$

for suitable polynomials q^i with algebraic coefficients. Thus

$$\liminf_{n \rightarrow \infty} \left| \sum_{i=1}^t g^i(n)x_0^i \right| = \liminf_{n \rightarrow \infty} \left| \sum_{i=1}^t q^i(1/n)\gamma_i^n + \overline{q^i(1/n)\gamma_i^n} \right|$$

We defer the proof of the following lemma, which requires tools from Diophantine analysis and the theory of reals, to the next section.

► **Lemma 22.** *Let $\gamma_1, \dots, \gamma_t$ be algebraic numbers with modulus 1. Let q^1, \dots, q^t be polynomials with algebraic coefficients. The quantity*

$$\liminf_{n \rightarrow \infty} \left| \sum_{i=1}^t q^i(1/n)\gamma_i^n + \overline{q^i(1/n)\gamma_i^n} \right|$$

can be effectively computed. If it is greater than zero, there is an effectively computable N satisfying the requirement of Lemma 21.

Proof of Theorem 7 (2). We are now ready to aggregate our case analysis into the proof the pseudo-reachability in hyperplanes is decidable. Given $A \in \mathbb{Q}^{m \times m}$, $x_0 \in \mathbb{Q}^m$ and $H = \{x : c^\top \cdot x = 0\}$, we first convert A to real Jordan normal form as described in Section 2 to obtain $J = Q^{-1}AQ$. We then perform a coordinate transform on x_0 and H to obtain $H' = \{x : c^\top Qx = 0\}$ and $x'_0 = Q^{-1}x_0$. The original problem is now equivalent to pseudo-reachability of H' from x'_0 under J .

Next, we remove dimensions from $x'_0, c^\top Q$ and J that do not correspond to relevant blocks and determine the relevant spectral radius $\rho_H(J)$ of J . If $\rho_H(J) = 1$ then H' is reachable by Lemma 19(1). If $\rho_H(J) < 1$, then by Lemma 19(2), H' is pseudo-reachable if and only if $\mathbf{0} \in H'$ or $x'_0, Jx'_0, \dots, J^N x'_0$ hits H' , where N is the computable bound in the Lemma.

Finally, we consider the case where $\rho_H(J) > 1$. Let J_1, \dots, J_t be the blocks of J with $\rho(J) = \rho_H(J)$ and $c^1, \dots, c^t, x_0^1, \dots, x_0^t$ be the corresponding coordinates of $c^\top Q$ and x'_0 , respectively. Finally, compute the value of $\liminf_{n \rightarrow \infty} \left| \sum_{i=1}^t g^i(n)x_n^i \right|$ using Lemma 22 and use Lemma 21 to either immediately conclude reachability or to compute the bound N and determine reachability by checking if $x'_0, Jx'_0, \dots, J^N x'_0$ hits H' .

5 Proof of Lemma 22

We now prove a generalization of Lemma 22. Let $\lambda_1, \dots, \lambda_m$ be algebraic numbers of modulus 1 and let p_1, \dots, p_m be polynomials with algebraic coefficients. Let n range over the natural numbers. We show how to effectively determine the value of $\liminf_{n \rightarrow \infty} |\sum_{j=1}^m p_j(1/n)\lambda_j^n|$. Moreover, if the value is strictly greater than 0, we show we can find an explicit bound Δ and $N \in \mathbb{N}$ such that for all $n > N$, we have $|\sum_{j=1}^m p_j(1/n)\lambda_j^n| > \Delta$. Lemma 22 follows as a special case.

We require some technical machinery from the theory of Diophantine approximations. We need the following theorem of Masser [13]. A proof can be found in [5] or [9].

► **Theorem 23** ([13]). *Let $m \in \mathbb{N}$ be fixed and let $\lambda_1, \dots, \lambda_m$ be complex algebraic numbers each of modulus 1. Consider the free Abelian group*

$$L = \{(v_1, \dots, v_m) \in \mathbb{Z}^m : \lambda_1^{v_1} \lambda_2^{v_2} \dots \lambda_m^{v_m} = 1\}.$$

L has a basis $\{\vec{\ell}_1, \dots, \vec{\ell}_p\} \subseteq \mathbb{Z}^m$ (with $p \leq m$), where the entries of each of the $\vec{\ell}_j$ are all polynomially bounded in the total description length of $\lambda_1, \dots, \lambda_m$. Moreover, such a basis can be also computed in time polynomial in the total description length.

Let L be as described in Theorem 23 above and suppose we have computed a basis $\{\vec{\ell}_1, \dots, \vec{\ell}_p\} \subseteq \mathbb{Z}^m$. For each $j \in \{1, \dots, p\}$, let $\vec{\ell}_j = (\ell_{j,1}, \dots, \ell_{j,m})$. Now we define a set

$$T := \{(z_1, \dots, z_m) \in \mathbb{C}^m : |z_1| = \dots = |z_m| = 1 \text{ and} \\ \text{for each } j \in \{1, \dots, p\}, z_1^{\ell_{j,1}} \dots z_m^{\ell_{j,m}} = 1\} \quad (8)$$

Notice that $|z| = 1 \iff \operatorname{Re}(z)^2 + \operatorname{Im}(z)^2 - 1 = 0$, and the $\ell_{j,k}$ are fixed integers, and thus the conditions above can be written as polynomials in the real and imaginary parts of z . Thus T is an algebraic set.

We now state a version of Kronecker's theorem on simultaneous Diophantine approximation. A derivation of this version of the theorem from the standard version ([10] Chap 23) can be found in [15].

► **Theorem 24** (Kronecker's theorem, density version). *Let T be defined from $\lambda_1, \dots, \lambda_m$ as in (8). Then $\{(\lambda_1^n, \dots, \lambda_m^n) : n \in \mathbb{N}\}$ is a dense subset of T .*

Theorem 24 enables us to compute the \liminf by minimizing a function over a compact algebraic set:

► **Theorem 25.** *Let $\lambda_1, \dots, \lambda_m$ be complex numbers of modulus 1. Let p_1, \dots, p_m be polynomials (with algebraic coefficients) with constant terms c_1, \dots, c_m respectively. Let $\mathbf{z} = (z_1, \dots, z_m)$ and $\mathbf{c} = (c_1, \dots, c_m)$. We have that*

$$\liminf_{n \rightarrow \infty} \left| \sum_{j=1}^m p_j(1/n)\lambda_j^n \right| = \liminf_{n \rightarrow \infty} \left| \sum_{j=1}^m c_j \lambda_j^n \right| = \inf_{\mathbf{z} \in T} |\mathbf{c}^\top \cdot \mathbf{z}| = \min_{\mathbf{z} \in T} |\mathbf{c}^\top \cdot \mathbf{z}|,$$

where T is the algebraic set computed in (8) as the closure of $\{(\lambda_1^n, \dots, \lambda_m^n) : n \in \mathbb{N}\}$.

To prove the theorem, we need the following lemma that shows that we can replace the polynomials by their constant terms.

► **Lemma 26.** Let $\lambda_1, \dots, \lambda_m$ be complex numbers of modulus 1. Let p_1, \dots, p_m be polynomials (with algebraic coefficients) with constant terms c_1, \dots, c_m respectively. Then

$$\liminf_{n \rightarrow \infty} \left| \sum_{j=1}^m p_j(1/n)\lambda_j^n \right| = \liminf_{n \rightarrow \infty} \left| \sum_{j=1}^m c_j \lambda_j^n \right|.$$

Proof. (of Theorem 25). The first equality follows from Lemma 26 and the second follows from Theorem 24. The third equality holds because the function $\mathbf{z} \mapsto |\mathbf{c}^\top \cdot \mathbf{z}|$ is continuous and T is compact. ◀

Now, since T is an algebraic set, the minimum $\min_{\mathbf{z} \in T} |\mathbf{c}^\top \cdot \mathbf{z}|$ can be expressed in the theory of reals with addition and multiplication (omitting the encoding of absolute values):

$$\exists \mathbf{z} \in T. v = |\mathbf{c}^\top \cdot \mathbf{z}| \wedge \forall \mathbf{z}' \in T. v \leq |\mathbf{c}^\top \cdot \mathbf{z}'|$$

Therefore, by Tarski's theorem [18, 2, 7], we can characterize the unique v that attains the minimum.

Suppose the minimum v is some number $B > 0$. In this case, we require a bound $\Delta \in \mathbb{R}$ and $N \in \mathbb{N}$ such that $|\sum_{j=1}^m p_j(1/n)\lambda_j^n| > B$ for all $n > N$. By emulating the proof of Lemma 26, we can find a bound N such that for all $n > N$, we have $|\sum_{j=1}^m p_j(1/n)\lambda_j^n| > B/2$. The required bounds are $\Delta = B/2$ and this N .

This concludes the proof of Lemma 22 and therefore also Theorem 7.

References

- 1 Dmitri V. Anosov. Geodesic flows on closed Riemannian manifolds of negative curvature. *Proc. Steklov Inst. Math.*, 90, 1967.
- 2 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2006.
- 3 Rufus Bowen. *Equilibrium States and the Ergodic Theory of Anosov Diffeomorphisms*, volume 470 of *Lecture Notes in Mathematics*. Springer-Verlag, 1975.
- 4 Jin-Yi Cai. Computing Jordan normal forms exactly for commuting matrices in polynomial time. *Int. J. Found. Comput. Sci.*, 5(3/4):293–302, 1994.
- 5 Jin-Yi Cai, Richard J. Lipton, and Yechezkel Zalcstein. The complexity of the A B C problem. *SIAM J. Comput.*, 29(6), 2000.
- 6 Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 1993.
- 7 George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Brakhage, editor, *Automata Theory and Formal Languages*, pages 134–183, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.
- 8 Charles C. Conley. *Isolated invariant sets and the Morse index*, volume 25 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1978.
- 9 Guoqiang Ge. *Algorithms Related to Multiplicative Representations of Algebraic Numbers*. PhD thesis, U.C. Berkeley, 1993.
- 10 Godfrey H. Hardy and Edward M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 1999.
- 11 Michael A. Harrison. Lectures on linear sequential machines. Technical report, DTIC Document, 1969.
- 12 Ravindran Kannan and Richard J. Lipton. Polynomial-time algorithm for the orbit problem. *J. ACM*, 33(4):808–821, 1986.
- 13 David W. Masser. Linear relations on algebraic groups. In *New Advances in Transcendence Theory*. Camb. Univ. Press, 1988.

- 14 Maurice Mignotte, Tarlok N. Shorey, and Robert Tijdeman. The distance between terms of an algebraic recurrence sequence. *J. für die reine und angewandte Math.*, 349, 1984.
- 15 Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 366–379, 2014.
- 16 Joël Ouaknine and James Worrell. On linear recurrence sequences and loop termination. *ACM SIGLOG News*, 2(2):4–13, 2015.
- 17 Christos H. Papadimitriou and Georgios Piliouras. From nash equilibria to chain recurrent sets: An algorithmic solution concept for game theory. *Entropy*, 20(10):782, 2018.
- 18 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. *J. Symb. Comp.*, 1992.
- 19 Eduardo D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer-Verlag, Berlin, Heidelberg, 1998.
- 20 Terence Tao. *Structure and randomness: pages from year one of a mathematical blog*. American Mathematical Society, 2008.
- 21 Nikolai K. Vereshchagin. The problem of appearance of a zero in a linear recurrence sequence (in russian). *Mat. Zametki*, 38(2), 1985.

A Proof of Proposition 8

We want to show

$$Q \tilde{\mathcal{O}}_{\gamma_2}(B, Q^{-1}x) \subseteq \tilde{\mathcal{O}}_{\epsilon}(A, x) \subseteq Q \tilde{\mathcal{O}}_{\gamma_1}(B, Q^{-1}x), \quad (9)$$

where $\gamma_1 = \epsilon \|Q^{-1}\|_{\infty}$ and $\gamma_2 = \epsilon / \|Q\|_{\infty}$.

Take any $y \in \tilde{\mathcal{O}}_{\epsilon}(A, x)$. We show that $Q^{-1}y \in \tilde{\mathcal{O}}_{\gamma_1}(B, Q^{-1}x)$ to get the right-hand side of (9). Since $y \in \tilde{\mathcal{O}}_{\epsilon}(A, x)$, there is a state trajectory (x_0, x_1, \dots) and a sequence (d_0, d_1, \dots) such that $x_0 = x$, $x_{n+1} = Ax_n + d_n$, $d_n \in [-\epsilon, \epsilon]^m$ for all $n \in \mathbb{N}$, and y appears in the state trajectory. We construct a new state trajectory (y_0, y_1, \dots) and the sequence $(\bar{d}_0, \bar{d}_1, \dots)$ with the transformation $x_n = Qy_n$ and $d_n = Q\bar{d}_n$. Then we have $y_{n+1} = Q^{-1}AQy_n + Q^{-1}d_n = By_n + \bar{d}_n$. Note that $\|\bar{d}_n\|_{\infty} = \|Q^{-1}d_n\|_{\infty} \leq \|Q^{-1}\|_{\infty} \|d_n\|_{\infty} \leq \gamma_1$. Since y appears in the state trajectory (x_0, x_1, \dots) , $Q^{-1}y$ appears in the state trajectory (y_0, y_1, \dots) with $y_0 = Q^{-1}x_0 = Q^{-1}x$. Therefore, $Q^{-1}y \in \tilde{\mathcal{O}}_{\gamma_1}(B, Q^{-1}x)$ which results in $y \in Q \tilde{\mathcal{O}}_{\gamma_1}(B, Q^{-1}x)$.

To prove the left-hand side of (9), We invoke the right-hand side by replacing (A, B, Q, x, ϵ) with $(B, A, Q^{-1}, Q^{-1}x, \gamma_2)$. This gives $\tilde{\mathcal{O}}_{\gamma_2}(B, Q^{-1}x) \subseteq Q^{-1} \tilde{\mathcal{O}}_{\gamma'_1}(A, x)$ with $\gamma'_1 = \gamma_2 \|Q\|_{\infty}$. Setting $\gamma'_1 = \epsilon$ proves the left-hand side of (9).

To prove that $\tilde{\mathcal{O}}(A, x) = Q \tilde{\mathcal{O}}(B, Q^{-1}x)$, we take intersection of all the sides in (9) over $\epsilon > 0$:

$$\bigcap_{\epsilon > 0} Q \tilde{\mathcal{O}}_{\gamma_2}(B, Q^{-1}x) \subseteq \bigcap_{\epsilon > 0} \tilde{\mathcal{O}}_{\epsilon}(A, x) \subseteq \bigcap_{\epsilon > 0} Q \tilde{\mathcal{O}}_{\gamma_1}(B, Q^{-1}x).$$

Due to the linear relation between γ_1 and γ_2 with ϵ , we get

$$Q \tilde{\mathcal{O}}(B, Q^{-1}x) \subseteq \tilde{\mathcal{O}}(A, x) \subseteq Q \tilde{\mathcal{O}}(B, Q^{-1}x) \quad \Rightarrow \quad \tilde{\mathcal{O}}(A, x) = Q \tilde{\mathcal{O}}(B, Q^{-1}x).$$

B Proofs from Section 3

B.1 Proof of Lemma 9

Any $t \in \tilde{\mathcal{O}}_\epsilon(A, s)$ is of the form $t = A^n s + \sum_{i=0}^{n-1} A^i d_{n-i-1}$ for some $n \in \mathbb{N}$ and some d_i with $\|d_i\|_\infty \leq \epsilon$. This means $s = A^{-n} t + \sum_{i=0}^{n-1} A^{-i} d'_{n-i-1}$ with $d'_{n-1-i} = A^{-1} d_i$. Since $\|d'_{n-1-i}\|_\infty \leq \|A^{-1}\|_\infty \epsilon$, we get $s \in \tilde{\mathcal{O}}_\gamma(A^{-1}, t)$. To get (4), notice that

$$t \in \tilde{\mathcal{O}}(A, s) \Rightarrow t \in \bigcap_{\epsilon > 0} \tilde{\mathcal{O}}_\epsilon(A, s) \Rightarrow s \in \bigcap_{\gamma > 0} \tilde{\mathcal{O}}_\gamma(A^{-1}, t) \Rightarrow s \in \tilde{\mathcal{O}}(A^{-1}, t).$$

Applying the same argument to the matrix A^{-1} will give the other side of (4).

B.2 Proof of Theorem 18

We show that S is pseudo-reachable from x_0 under A if and only if there exists $x \in \bar{S}$ that is pseudo-reachable from x_0 under A , allowing us to restrict our attention to compact sets and the existence of a pseudo-reachable point in a set as opposed to pseudo-reachability of the set as a whole. Deciding pseudo-reachability then reduces to checking whether $\bar{S} \cap \tilde{\mathcal{O}}(J, x_0) = \emptyset$, which can be computed using Lemma 17.

Suppose S is pseudo-reachable. Let $(\epsilon_i)_{i \in \mathbb{N}}$ be a sequence of positive numbers with $\lim_{\epsilon \rightarrow 0} \epsilon_i = 0$, and $(x_i)_{i \in \mathbb{N}}$ be a sequence of elements of S such that x_i is ϵ_i -pseudo-reachable for all $i \geq 0$. By the Bolzano–Weierstrass theorem, boundedness of S implies that $(x_i)_{i \in \mathbb{N}}$ must have a limit point x in \bar{S} . To argue that x is pseudo-reachable, let $\epsilon > 0$. Since x is the limit point of $(x_i)_{i \in \mathbb{N}}$, there must exist an $\frac{\epsilon}{2}$ -pseudo-orbit $(y_i)_{i \in \mathbb{N}}$ containing a point y_N such that $\|x - y_N\|_\infty < \frac{\epsilon}{2}$. Therefore, x is ϵ -pseudo-reachable from s via the sequence $s, y_1, \dots, y_{N-1}, x$.

Now suppose $x \in \bar{S}$ is pseudo-reachable. To argue that S is pseudo-reachable, let $\epsilon > 0$. Since $x \in \bar{S}$, there must exist a point $x' \in S$ such that $\|x' - x\|_\infty < \frac{\epsilon}{2}$. Since x is $\frac{\epsilon}{2}$ -pseudo-reachable, x' must be ϵ -pseudo-reachable.

C Proof of Lemma 26

We can write $p_j(1/n)$ as $c_j + \sum_{i=1}^{d_j} c_{(j,i)} \frac{1}{n^i}$, where c_j is the constant term, $c_{(j,i)}$ are the other coefficients, and d_j is the degree. Define $A_j = \sum_{i=1}^{d_j} |c_{(j,i)}|$ and observe that

$$|p_j(1/n) - c_j| < \left| \sum_{i=1}^{d_j} c_{(j,i)} \frac{1}{n^i} \right| < \frac{\sum_{i=1}^{d_j} |c_{(j,i)}|}{n} = \frac{A_j}{n}$$

Thus for any ϵ , setting $N_j(\epsilon) = \lceil A_j/\epsilon \rceil$ ensures that

$$n > N_j(\epsilon) \implies |p_j(1/n) - c_j| < \epsilon.$$

Define $N(\epsilon) = \max_{j \in \{1, \dots, m\}} N_j(\epsilon/m)$.

▷ **Claim 27.** Let S_n be defined as $|\sum_{j=1}^m c_j \lambda_j^n|$. For all $\epsilon > 0$,

$$S_n - \epsilon \leq \left| \sum_{j=1}^m p_j(1/n) \lambda_j^n \right| \leq S_n + \epsilon$$

Taking the limit inferior of each term gives us the desired result.

34:20 The Pseudo-Skolem Problem is Decidable

Proof of Claim 27. We write

$$\left| \sum_{j=1}^m p_j(1/n) \lambda_j^n \right| = \left| \sum_{j=1}^m (c_j + p_j(1/n) - c_j) \lambda_j^n \right|,$$

which gives us

$$S_n - \left| \sum_{j=1}^m (p_j(1/n) - c_j) \lambda_j^n \right| \leq \left| \sum_{j=1}^m p_j(1/n) \lambda_j^n \right| \leq S_n + \left| \sum_{j=1}^m (p_j(1/n) - c_j) \lambda_j^n \right|$$

and thus

$$S_n - \sum_{j=1}^m |(p_j(1/n) - c_j) \lambda_j^n| \leq \left| \sum_{j=1}^m p_j(1/n) \lambda_j^n \right| \leq S_n + \sum_{j=1}^m |(p_j(1/n) - c_j) \lambda_j^n|,$$

by elementary properties of sums of absolute values. Observing that λ_j s have absolute value 1, we can reduce the proposition above to

$$S_n - \sum_{j=1}^m |(p_j(1/n) - c_j)| \leq \left| \sum_{j=1}^m p_j(1/n) \lambda_j^n \right| \leq S_n + \sum_{j=1}^m |(p_j(1/n) - c_j)|.$$

Now setting $n > N(\epsilon) = \max_{j \in \{1, \dots, m\}} N_j(\epsilon/m)$, we have $|(p_j(1/n) - c_j)| < \epsilon/m$ for all j , which gives us

$$S_n - \epsilon \leq \left| \sum_{j=1}^m p_j(1/n) \lambda_j^n \right| \leq S_n + \epsilon \quad \triangleleft$$

D Computing real JNF in polynomial time

We discuss how to compute the the real Jordan normal form of A in polynomial time. First compute, in polynomial time, the (complex) Jordan normal form J' and matrices T, T^{-1} such that $A = T J' T^{-1}$ using the algorithm from [4].

Computing J . Suppose, without loss of generality, that

$$J' = \text{diag}(J'_1, J'_2, \dots, J'_{2k-1}, J'_{2k}, J'_{2k+1}, \dots, J'_{2k+z})$$

where for $1 \leq j \leq k$, the Jordan blocks J'_{2j-1} and J'_{2j} have the same dimension and have conjugate eigenvalues $\lambda_j = a_j + b_j i$ and $\bar{\lambda} = a_j - b_j i$, respectively. The blocks $J'_{2k+1}, \dots, J'_{2k+z}$, on the other hand, have real eigenvalues. J is obtained by replacing, for each $1 \leq j \leq k$, $\text{diag}(J'_{2j-1}, J'_{2j})$ with a real Jordan block of the same dimension with $\Lambda = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$ and keeping the blocks $J'_{2k+1}, \dots, J'_{2k+z}$ unchanged.

Computing P . Let $\kappa(j)$ denote the multiplicity of the Jordan block J'_i for $1 \leq i \leq 2k+z$, and $v_1^1, \dots, v_{\kappa(1)}^1, \dots, v_1^{2k}, \dots, v_{\kappa(2k)}^{2k}, \dots, v_1^{2k+z}, \dots, v_{\kappa(2k+z)}^{2k+z} \in \overline{\mathbb{Q}}^m$ be the columns of T . It will be the case that for all $1 \leq j \leq k$ and l , $v_l^{2j-1} = v_l^{2j}$ in the sense that $v_l^{2j-1} = x_l^j + y_l^j i$ and $v_l^{2j} = x_l^j - y_l^j i$ for vectors $x_l^j, y_l^j \in \mathbb{R}^m$. Moreover, for $j > 2k$, $v_l^{2j} \in \mathbb{R}^m$. Finally, columns of P are obtained from columns of T as follows. For $1 \leq j \leq k$ and all l , replace v_l^{2j-1} with x_l^j and v_l^{2j} with y_l^j and keep v_l^{2k+z} for all l and $m > 0$ unchanged, in the same way the proof of existence of real Jordan normal form proceeds.

Computing P^{-1} . Summarizing the construction above, P is obtained from T by replacing columns $x + yi$ and $x - yi$, $x, y \in \mathbb{R}^m$ by x and y , respectively. Since $x = \frac{1}{2}(x + yi) + \frac{1}{2}(x - yi)$ and $y = -\frac{1}{2}i(x + yi) + \frac{1}{2}i(x - yi)$, this construction is linear and we can write $P = T \cdots A$ for some $A \in \mathbb{C}^{m \times m}$ with entries in $\{\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}i, -\frac{1}{2}i, 1, 0\}$. Moreover, the linear transformation is clearly invertible: $x + yi = 1 \cdot x + iy$ and $x - yi = 1 \cdot x - (-i)y$, and hence $A^{-1} \in \mathbb{C}^{m \times m}$ with entries in $\{1, i, -i\}$. Finally, compute P^{-1} via $P = TA \implies P^{-1} = A^{-1}T^{-1}$, observing that we already know how to compute T^{-1} in polynomial time.

A Recursion-Theoretic Characterization of the Probabilistic Class PP

Ugo Dal Lago 

University of Bologna, Italy
INRIA Sophia Antipolis, France

Reinhard Kahle 

Carl Friedrich von Weizsäcker Center, Universität Tübingen, Germany
Center for Mathematics and Applications (CMA), FCT, Nova University Lisbon, Caparica, Portugal

Isabel Oitavem¹ 

Center for Mathematics and Applications (CMA), FCT, Nova University Lisbon, Caparica, Portugal
Department of Mathematics, FCT, Nova University Lisbon, Caparica, Portugal

Abstract

Probabilistic complexity classes, despite capturing the notion of feasibility, have escaped any treatment by the tools of so-called implicit-complexity. Their inherently semantic nature is of course a barrier to the characterization of classes like BPP or ZPP, but not all classes are semantic. In this paper, we introduce a recursion-theoretic characterization of the probabilistic class PP, using recursion schemata with pointers.

2012 ACM Subject Classification Theory of computation → Recursive functions

Keywords and phrases Implicit complexity, tree-recursion, probabilistic classes, polynomial time, PP

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.35

Funding *Ugo Dal Lago*: Research supported by the ERC CoG DIAPASoN, GA 818616, and by the ANR Project PPS 19CE480014.

Reinhard Kahle: Research supported by the Udo Keller Foundation and as for Isabel Oitavem.

Isabel Oitavem: National funds through the FCT – Fundação para a Ciência e Tecnologia, I.P., under the scope of the project UIDB/00297/2020 (Center for Mathematics and Applications).

1 Introduction

Implicit computational complexity aims at introducing and studying characterizations of complexity classes by recursion theory, proof theory, and programming language tools. This has allowed to shed some light on the impact of programming and recursion schemes to resource consumption. As an example, nested recursion on notation is known to be harmful to time complexity, and thus needs to be appropriately controlled if one wants to stay within the realm of polynomial time computable functions, as shown by Bellantoni-Cook and Leivant in their works on safe and tiered recursion [2, 9].

If one wants to go beyond polynomial time, a way of enriching recursion schemes without breaking the correspondence with (relatively small) complexity classes is the use of recursion schemes based on *pointers* [3], which leads to characterizations of NP and FPSPACE (the class of *functions* corresponding to PSPACE), [14, 13].

One family of complexity classes between polytime and polyspace which has so far escaped any implicit treatment are the probabilistic ones. Our goal in this paper is precisely the one of exploring the potential of pointers in recursion-theoretic contexts as a tool to characterize

¹ corresponding author



probabilistic classes of computational complexity. In this work we study PP, the class of decision problems solvable by probabilistic Turing machines in polynomial time with an error probability of less than $\frac{1}{2}$ for all instances. The class PP was originally defined by Gill in [6]. It is well-known that PP contains NP and that it is contained in PSPACE; it is open whether these inclusions are proper or not.

Our strategy consists in extending the existing recursion scheme for NP going towards the one for FSPACE, but without going too far. Surprisingly, this is possible despite the inherently quantitative nature of PP's acceptance criterion. This way, we obtain the first purely recursion-theoretic characterization of the probabilistic class PP.

This characterization is described in two stages, \mathbf{ST}_P and \mathbf{ST}_{PP} , where \mathbf{ST}_P characterizes the class of functions computable in polynomial time by deterministic Turing machines, FPTIME – see [2]. \mathbf{ST}_{PP} results then from “strengthening” \mathbf{ST}_P with a scheme designed to characterize the decision problems of PP. That scheme is the tree-recursion scheme of FSPACE [13], but with a fixed step function. Therefore, the characterization of the class PP given here is aligned with the existing recursion-theoretic characterizations of FPTIME and FSPACE.

2 Algebras, Functions, and Complexity Classes

In this section, we introduce some preliminary concepts that will accompany us in the rest of the paper. In particular, we will show how binary strings and functions over them allow us to capture standard, deterministic classes like FPTIME and FSPACE.

2.1 Algebras and Recursion Schemes

Let us consider the word algebra \mathbb{W} , i.e. the algebra generated by one nullary and two unary constructors, respectively indicated as ϵ , \mathbf{S}_0 and \mathbf{S}_1 . The algebra \mathbb{W} can naturally be interpreted over the set $\{0, 1\}^*$ of all binary words. We abbreviate \mathbf{S}_0x and \mathbf{S}_1x as $x0$ and $x1$, respectively. We consider a predecessor symbol \mathbf{P} of arity 1, and a conditional function symbol \mathbf{C} of arity 4. They are defined as follows: $\mathbf{P}(\epsilon) = \epsilon$, $\mathbf{P}(xi) = x$ and $\mathbf{C}(\epsilon, x, y_0, y_1) = x$, $\mathbf{C}(zi, x, y_0, y_1) = y_i$, $i \in \{0, 1\}$.

Recursion schemes on the algebra \mathbb{W} can be built in two different ways:

1. First of all, one can proceed by *recursion on notation*, namely by building a function f out of g and h by stipulating that $f(\epsilon, \bar{x}) = g(\epsilon, \bar{x})$ and $f(zi, \bar{x}) = h(zi, \bar{x}, f(z, \bar{x}))$, where $i \in \{0, 1\}$;
2. Secondly, one can also build f through *tree-recursion with pointers* from g and h , namely by $f(p, \epsilon, \bar{x}) = g(p, \epsilon, \bar{x})$ and $f(p, zi, \bar{x}) = h(p, zi, \bar{x}, f(p0, z, \bar{x}), f(p1, z, \bar{x}))$, $i \in \{0, 1\}$.

In both the recursion schemes above, g is designated as the *base function* and h as the *step function*. In tree-recursion, the first input of f is called the *pointer*. Noticeably, the characterization of PP presented here results from considering the scheme (2) with a fixed step function. Informally, the step function that we fix is a FPTIME function corresponding to binary addition (we add the number of accepting configurations), but with some nuances in order to capture the probabilistic class PP.

Above and along the paper, \bar{x} abbreviates x_1, \dots, x_n for some natural number n . Moreover, $|\bar{x}| = (|x_1|, \dots, |x_n|)$ where, for $1 \leq k \leq n$, $|x_k|$ denotes the length of x_k , i.e. the number of \mathbf{S}_0 and \mathbf{S}_1 in x_k . We extend these notations to function symbols.

2.2 Capturing Polynomial Time and Space: \mathbf{ST}_P and $\mathbf{ST}_{FPSPACE}$

If either recursion on notation or tree recursion are applied without any restriction, there is no hope to capture interesting complexity classes: in either case we end up capturing the whole class of primitive recursion. From now on, then, we adopt the framework introduced by Bellantoni-Cook [2], which has proved to be quite useful in appropriately restricting function algebras so as to capture small complexity classes. In particular, function terms have two sorts of input positions, dubbed *normal* and *safe*. As is customary, we write normal and safe input positions by this order, separated by semicolon, e.g. in the expression $f(\bar{x}; \bar{y})$, the parameters in \bar{x} are normal while those in \bar{y} are safe.

We are now in a position to define the three closure operators which we will employ in the following, and which will be

► **Definition 1** (Closure Operators). *Let g, h, \bar{r} and \bar{s} be sorted functions. The following are three ways of building a new sorted function term out of (some of) them.*

■ **Sorted Composition**, SC , by which we can build the sorted function f such that

$$f(\bar{x}; \bar{y}) = h(\bar{r}(\bar{x}); \bar{s}(\bar{x}; \bar{y}))$$

■ **Sorted Recursion on Notation**, SR , which allows us to derive the sorted function f by a recursion scheme:

$$f(\epsilon, \bar{x}; \bar{y}) = g(\epsilon, \bar{x}; \bar{y})$$

$$f(z0, \bar{x}; \bar{y}) = h(z0, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y}))$$

$$f(z1, \bar{x}; \bar{y}) = h(z1, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y})).$$

■ **Sorted Tree-Recursion**, STR , which derives the sorted function f , again by recursion

$$f(p, \epsilon, \bar{x}; \bar{y}) = g(p, \epsilon, \bar{x}; \bar{y})$$

$$f(p, z0, \bar{x}; \bar{y}) = h(p, z0, \bar{x}; \bar{y}, f(p0, z, \bar{x}; \bar{y}), f(p1, z, \bar{x}; \bar{y}))$$

$$f(p, z1, \bar{x}; \bar{y}) = h(p, z1, \bar{x}; \bar{y}, f(p0, z, \bar{x}; \bar{y}), f(p1, z, \bar{x}; \bar{y}))$$

A few observations about the two recursion operators are helpful now. Please note that, according to the semi-colon discipline, both the *pointer* p and the *recursion variable* z are in normal input positions, while the results of *recursive calls* go into safe input positions. This, in particular, prevents the results of recursive calls to be fed to a function itself defined by recursion on that same variable, ultimately avoiding a blowup in size and complexity.

► **Definition 2** (Function Algebras). *Let \mathcal{I} be the class of function terms including the constructors of \mathbb{W} (i.e. the functions $\epsilon, \mathbf{S}_0, \mathbf{S}_1$), the predecessor \mathbf{P} , the conditional \mathbf{C} , and the projection functions (over both input sorts).*

1. \mathbf{ST}_P is the closure of \mathcal{I} under SC and SR ;
2. $\mathbf{ST}_{FPSPACE}$ is the closure of \mathcal{I} under SC and STR .

As known results one has that:

► **Proposition 3.**

1. \mathbf{ST}_P characterizes $FPTIME$ and $\mathbf{ST}_{FPSPACE}$ characterizes $FPSPACE$ – see, respectively, [2] and [13];
2. $\mathbf{ST}_P \subseteq \mathbf{ST}_{FPSPACE}$, as classes of input-sorted function terms.

Notice that in this framework one can run a recursion over an output of a function which is itself defined by recursion. There are some other properties concerning \mathbf{ST}_P that we use in the paper. We summarize them here for further reference – see [2] and [14], Remark 2.

► Remark 4.

1. Let $f(\bar{x})$ be a polytime function. Then $f(\bar{x};)$ is in \mathbf{ST}_P .
2. For any polynomial (with natural coefficients) q , there exists a term $t \in \mathbf{ST}_P$ such that $\forall \bar{x} \ q(|\bar{x}|) = |t(\bar{x};)|$;
3. For any polytime function F there exist a function term f , in \mathbf{ST}_P , and a monotone polynomial q_F such that $\forall \bar{w} \forall y \ |y| \geq q_F(|\bar{w}|) \Rightarrow F(\bar{w}) = f(y; \bar{w})$.

3 The Algebra \mathbf{ST}_{PP}

In this section, we define a function algebra \mathbf{ST}_{PP} , based on \mathbf{ST}_P and on the tree-recursion scheme with pointers. The main idea behind \mathbf{ST}_{PP} is to constrain $\mathbf{ST}_{FPSPACE}$ in such a way that tree-recursion is restricted to a very specific step function. The next section is devoted to introducing and motivating this function.

3.1 On the \boxplus Function

In the following, we will extensively work with the following FPTIME functions:

- The unary function *read* which, on input w , returns 10 if the last bit of w is 1, and 0 otherwise;
- Addition on binary words *seen as* natural numbers, indicated as $+$. It results from considering the following

0	1	10	11	100	...	(binary words)
↓	↓	↓	↓	↓		
0	1	2	3	4	...	(natural numbers)

and the usual addition over \mathbb{N} , extending it to all binary words assuming that the empty string ϵ and words starting with 0 all correspond to $0 \in \mathbb{N}$. For instance, $+(10, 10) = 100$, while $+(00, 10) = 10$. We use infix notation for $+$;

- The binary function \oplus such that $\oplus(w, v)$ is $read(w) + read(v)$, for which we also use the infix notation. As an example, $01 \oplus 100 \oplus 1 = read(01) + read(100) + read(1) = 10 + 0 + 10 = 100$. The function \oplus only depends on the last bit of its two arguments. Notice that any finite sum $\sum_n read(w_n)$ is always a binary word ending with 0, because for every such binary word w it holds that $read(w)$ is 0 or 10;
- The binary function $\#$ defined as follows:

$$\#(z, w) = \begin{cases} 1 & \text{if } w > 2^{|z|} \\ 0 & \text{otherwise,} \end{cases}$$

where $2^{|z|}$ is the binary representation of the natural number $2^{|z|}$, i.e., it holds that $2^{|\epsilon|} = 1$ and that $2^{|z|} = \mathbf{S}_0(2^{|z|})$.

We reserve the symbol $+$ for the binary addition as described above. Whenever needed, we use $+\mathbb{N}$ to denote the usual addition over the natural numbers. The “greater than” relation defined over binary words ordered by length and, within the same length, lexicographically is indicated as $>$;

Finally, let the function \boxplus be a polytime function satisfying the following equations (where i ranges over $\{0, 1\}$):

$$\begin{aligned} \boxplus(\epsilon, i, w_0, w_1) &= \#(i, w_0 \oplus w_1), \\ \boxplus(p, i, w_0, w_1) &= w_0 \oplus w_1, && \text{if } p \neq \epsilon \\ \boxplus(\epsilon, zi, w_0, w_1) &= \#(zi, w_0 + w_1), && \text{if } z \neq \epsilon, \\ \boxplus(p, zi, w_0, w_1) &= w_0 + w_1, && \text{if } p, z \neq \epsilon, \end{aligned}$$

The conditions imposed now will become clear later. \boxplus is a polytime function. Therefore Remark 4(1) ensures the existence of a term in \mathbf{ST}_P – let us reuse the symbol \boxplus to denote it – such that $\boxplus(p, z, w_0, w_1;) = \boxplus(p, z, w_0, w_1)$. The input-sorted function $\boxplus \in \mathbf{ST}_P$ is used as step function in the tree-recursion scheme.

3.2 The Algebra \mathbf{ST}_{PP}

It is now time to introduce the function algebra which constitutes the main contribution of this paper, and which will be proved to characterize the class \mathbf{PP} in the coming section.

- **Definition 5.** \mathbf{ST}_{PP} is the closure of \mathbf{ST}_P under \mathbf{SC}_P and $\mathbf{TR}[\boxplus]$, where
- **Restricted sorted composition**, \mathbf{SC}_P , allows to build a sorted function f out of h and $\bar{r}, \bar{s} \in \mathbf{ST}_P$ as follows:

$$f(\bar{x}; \bar{y}) = h(\bar{r}(\bar{x}); \bar{s}(\bar{x}; \bar{y}))$$

- **Tree-recursion with step function** \boxplus derives the function f out of $g \in \mathbf{ST}_P$ as follows:

$$\begin{aligned} f(p, \epsilon, \bar{x};) &= g(p, \epsilon, \bar{x};) \\ f(p, z0, \bar{x};) &= \boxplus(p, z0, f(p0, z, \bar{x};), f(p1, z, \bar{x};);) \\ f(p, z1, \bar{x};) &= \boxplus(p, z1, f(p0, z, \bar{x};), f(p1, z, \bar{x};);) \end{aligned}$$

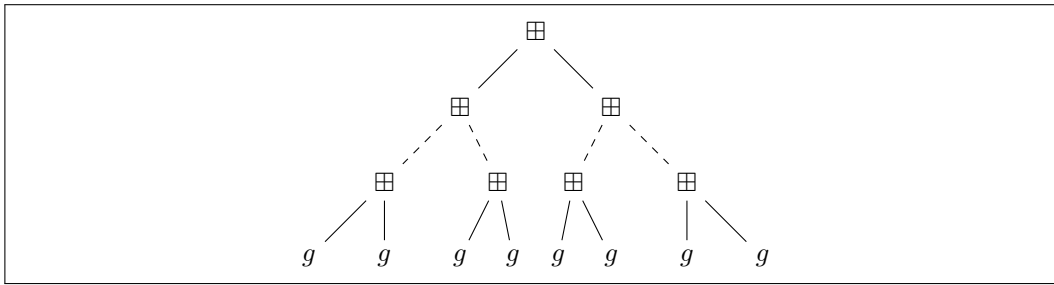
As already noticed, under this input-sorted discipline recursive calls are usually taken as safe arguments of the step function. However, that becomes irrelevant when the recursion scheme imposes a fixed step function – like in $\mathbf{TR}[\boxplus]$. The definition of any function term in \mathbf{ST}_{PP} involves at most one application of $\mathbf{TR}[\boxplus]$. This is a consequence of having the base function of $\mathbf{TR}[\boxplus]$ – g – and the inner functions of \mathbf{SC}_P – \bar{r} and \bar{s} – all in \mathbf{ST}_P .

It is known that $P \subseteq PP \subseteq \mathbf{PSPACE}$. On the term system side we observe that the correspondent inclusions are preserved.

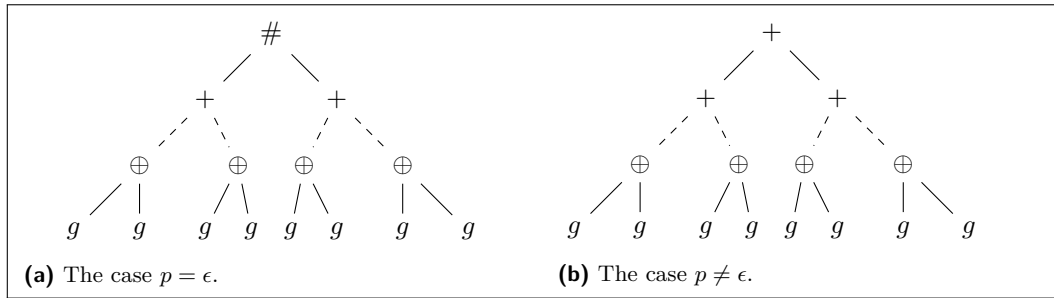
- **Remark 6.** $\mathbf{ST}_P \subseteq \mathbf{ST}_{PP} \subseteq \mathbf{ST}_{\mathbf{FPSPACE}}$, as classes of input-sorted function terms. The first inclusion is obvious, the second makes use of items (2) and (3) of Remark 4.

In the rest of this section, we will explain how the $\mathbf{TR}[\boxplus]$ scheme somehow mimics the acceptance condition underlying \mathbf{PP} . Suppose a function is obtained through $\mathbf{TR}[\boxplus]$ from a base function g ; then its evaluation on an input $(p, z, \bar{x};)$, for $z \neq \epsilon$ gives rise to a tree similar to that in Figure 1, where the pointer p and the recursion input z are omitted. By using them (the first and the second inputs of \boxplus), one is able to obtain different outputs for \boxplus depending on the level it occurs in the tree above. Therefore, according to the definition of \boxplus , one can distinguish two different situations.

- If f is evaluated on arguments in the form $(\epsilon, z, \bar{x};)$, then the tree from Figure 1 becomes the one in Figure 2(a). Again, inputs are omitted. One reads from the leaves, and by performing binary addition at all internal nodes one brings up to the root of the tree the



■ **Figure 1** The unfolding of $f(p, z, \bar{x})$.



■ **Figure 2** The unfolding of $f(p, z, \bar{x})$, depending on the value of p .

information about how many times 1 occurs at the leaves. Notice that the function #, after performing the binary addition operation, returns 1 if the sum meets the threshold (i.e. if strictly more than half of the leaves are labeled by 1) and 0 otherwise.

- If f is evaluated on arguments in the form (p, z, \bar{x}) , where $p \neq \epsilon$, then the tree becomes the one in Figure 2(b). The pointer increases along the paths of the tree. So, the first input of \boxplus does not assume the value ϵ . Therefore, the test function # is not called. Recall that \oplus reads the last bit of the leaves, coding the 1's by 10 (i.e. by the binary representation of 2). It should be clear that all nodes are labeled by strings ending by 0 (because they are binary representation of even numbers). In particular, the value returned to the root of the tree is a 0-1 word ending by 0.

4 ST_{PP} Characterizes PP

In this section, we prove that the algebra ST_{PP} we introduced in the previous section indeed characterizes our target class, namely PP.

We adopt the definition of PP given in [1]. In particular, we consider non-deterministic Turing machines as the underlying model of computation, and we make the assumption that every step of the computation can be made in *exactly two* possible ways. Thus, in the course of the computation, every configuration of the machine has exactly two next configurations. Machines are “clocked” by some constructible function, and the number of steps in each computation is exactly the number of steps allowed by the clock. If a final state is reached before this number of steps, then the computation is continued, doing nothing up to this number of steps. Moreover, every computation ends in a final state, which can be either ACCEPT or REJECT. A *probabilistic Turing machine* M , PTM for short, is a non-deterministic Turing machine as above in which acceptance is defined quantitatively rather than logically: the input \bar{x} is accepted if, and only if, *more than half* of the computations of M on \bar{x} end in the ACCEPT final state.

PP is the class of boolean functions (or languages) computed (resp. accepted) by polynomially clocked PTMs. While PP is a class of two valued functions, \mathbf{ST}_{PP} corresponds to a class of functions whose values can range over binary words. As a consequence, our correspondence theorem will say that $\mathcal{B}(\mathbf{ST}_{\text{PP}})$ coincides with PP, where $\mathcal{B}(\mathbf{ST}_{\text{PP}})$ denotes the boolean part of \mathbf{ST}_{PP} .

4.1 The Lower Bound

In this subsection, we will prove that \mathbf{ST}_{PP} is powerful enough to capture the behavior of any polynomially clocked probabilistic Turing machine. This proves that \mathbf{ST}_{PP} is at least as expressive as our target complexity class.

► **Lemma 7.** *Polynomially clocked PTMs can be simulated by functions in \mathbf{ST}_{PP} .*

Proof. Let M be a PTM which is clocked by some polynomial q (on the length of the input). We are going to simulate M by \mathbf{ST}_{PP} function terms. Let us assume, without any loss of generality, that machine configurations are encoded by binary words so that codes end by the code of the respective state. Codes of accepting final states end by 1 and any other state ends by 0. One may assume that, for a given input \bar{x} , all the configuration codes have the same length, $l(|\bar{x}|)$, which is polynomial on $|\bar{x}|$. Notice that all non-terminating configurations have two successor configurations. Therefore, we can split the transition function δ of M into two δ_0 and δ_1 . Let c be a \mathbf{ST}_{PP} function such that $c(\bar{x};)$ is the code of the initial configuration. Similarly, let t be a \mathbf{ST}_{PP} function such that $|t(\bar{x};)| = q(|\bar{x}|)$. (That t can be defined in \mathbf{ST}_{PP} come from the fact that q is a polynomial, cf. Remark 4(2)). For $i \in \{0, 1\}$, one may consider polytime computable functions Δ_i which, for a given configuration code w , return the next configuration code according to δ_i , or return w itself if there is no next configuration according to δ_i . Thus, by Remark 4(3), there exists a function term $\hat{\Delta}_i$ in \mathbf{ST}_{P} and a polynomial q_{Δ_i} such that $\forall w \forall y \ |y| \geq q_{\Delta_i}(|w|) \Rightarrow \Delta_i(w) = \hat{\Delta}_i(y; w)$. Replacing, in the previous expression, y by $L_{\Delta_i}(\bar{x};)$ one has that $\forall w \forall \bar{x} \ |L_{\Delta_i}(\bar{x};)| \geq q_{\Delta_i}(|w|) \Rightarrow \Delta_i(w) = \hat{\Delta}_i(L_{\Delta_i}(\bar{x};); w)$, where L_{Δ_i} is a \mathbf{ST}_{P} term as follows. Given an input \bar{x} , all configuration codes w satisfy $|w| = l(|\bar{x}|)$ where l is polynomial in $|\bar{x}|$. Thus, $q_{\Delta_i}(|w|)$ is equal to $(q_{\Delta_i} \circ l)(|\bar{x}|)$. The composition of polynomials is a polynomial, and so $q_{\Delta_i} \circ l$ is a polynomial in $|\bar{x}|$. Therefore, by Remark 4(2), there exists a function term L_{Δ_i} in \mathbf{ST}_{P} such that $|L_{\Delta_i}(\bar{x};)| = (q_{\Delta_i} \circ l)(|\bar{x}|)$. Now, recalling that $q_{\Delta_i}(|w|)$ is $(q_{\Delta_i} \circ l)(|\bar{x}|)$, one has $|L_{\Delta_i}(\bar{x};)| \geq q_{\Delta_i}(|w|)$ (thus, a fortiori $|L_{\Delta_i}(\bar{x};)| \geq q_{\Delta_i}(|w|)$). Therefore, for any input \bar{x} , given a configuration code w , $\Delta_i(w) = \hat{\Delta}_i(L_{\Delta_i}(\bar{x};); w)$ where $\hat{\Delta}_i$ and L_{Δ_i} are in \mathbf{ST}_{P} . This means that (reusing the symbol Δ_i) we can consider a function term $\Delta_i(\bar{x}; w) = \hat{\Delta}_i(L_{\Delta_i}(\bar{x};); w)$ in \mathbf{ST}_{P} , which for any input \bar{x} and a given configuration code w returns the next configuration code according to δ_i . Let us define an auxiliary sorted function, called *RUN*. *RUN* is defined, in \mathbf{ST}_{P} , by SR. For a path p and a (initial) configuration code $c(\bar{x};)$, *RUN* simulates the (sequential) computation performed by M along the branch p starting with the configuration code $c(\bar{x};)$.

$$\begin{aligned} \text{RUN}(\epsilon, \bar{x};) &= c(\bar{x};) \\ \text{RUN}(p0, \bar{x};) &= \Delta_0(\bar{x}; \text{RUN}(p, \bar{x};)) \\ \text{RUN}(p1, \bar{x};) &= \Delta_1(\bar{x}; \text{RUN}(p, \bar{x};)) \end{aligned}$$

Let us consider the function f defined by $\text{TR}[\boxplus]$ in \mathbf{ST}_{PP} :

$$\begin{aligned} f(p, \epsilon, \bar{x};) &= \text{RUN}(p, \bar{x};) \\ f(p, z0, \bar{x};) &= \boxplus(p, z0, f(p0, z, \bar{x};), f(p1, z, \bar{x};);) \\ f(p, z1, \bar{x};) &= \boxplus(p, z1, f(p0, z, \bar{x};), f(p1, z, \bar{x};);). \end{aligned}$$

By construction, one has that $M(\bar{x}) = f(\epsilon, t(\bar{x};), \bar{x};)$. ◀

As a consequence, one can prove the following.

► **Proposition 8.** *PP is contained in $\mathcal{B}(\mathbf{ST}_{PP})$.*

Proof. Any boolean function in PP can by definition be computed by a polynomially clocked PTM. It is thus immediate to conclude, from the previous lemma, that PP is contained in $\mathcal{B}(\mathbf{ST}_{PP})$. ◀

Before moving to the dual result, it is instructive to take a further look at how Lemma 7 is proved. Actually, closure of \mathbf{ST}_{PP} by $\text{TR}[\boxplus]$ is exploited just once. However, that single use of $\text{TR}[\boxplus]$ takes a non-boolean function as base function. So, although PP is a class of boolean functions, the non-boolean functions of the algebra play a crucial role in the characterization. To show that the boolean functions of the algebra, whose definitions involve $\text{TR}[\boxplus]$ with arbitrary polytime base functions, remain within the complexity class PP is nontrivial, and is essentially what we are going to prove in the next section.

4.2 The Upper Bound

Knowing that \mathbf{ST}_P characterizes FPTIME, it is clear that $\mathcal{B}(\mathbf{ST}_P)$ – which corresponds to P, the class of the polytime boolean-functions – is contained in PP. But how about the inclusion between $\mathcal{B}(\mathbf{ST}_{PP})$ and PP? This is precisely what we are going to prove in this section.

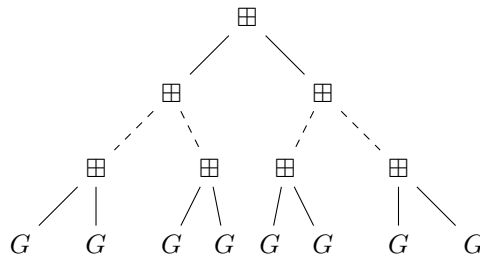
We first of all need the following lemma, which tells us, essentially, that $\text{TR}[\boxplus]$ makes sense from the point of view of the class PP. In this statement only boolean base functions are considered.

► **Lemma 9.** *Let G be in P and let F be a function defined as follows:*

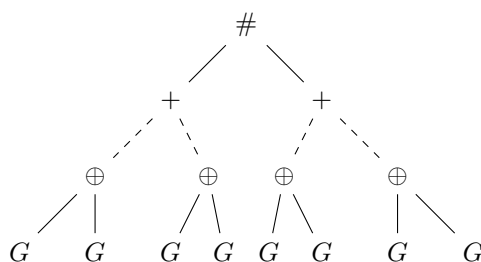
$$\begin{aligned} F(p, \epsilon, \bar{x}) &= G(p, \epsilon, \bar{x}) \\ F(p, z0, \bar{x}) &= \boxplus(p, z0, F(p0, z, \bar{x}), F(p1, z, \bar{x})) \\ F(p, z1, \bar{x}) &= \boxplus(p, z1, F(p0, z, \bar{x}), F(p1, z, \bar{x})). \end{aligned}$$

Then the function F_ϵ defined as $F_\epsilon(z, \bar{x}) = F(\epsilon, z, \bar{x})$, is in PP. Moreover, if G is computable in time t_G (on the sum of the length of its inputs), then $F(\epsilon, z, \bar{x})$ is clocked by $c \cdot |z| +_{\mathbb{N}} t_G(|p| +_{\mathbb{N}} |z| +_{\mathbb{N}} \sum_i |x_i|) +_{\mathbb{N}} 1$, for some constant c .

Proof. In order to compute F_ϵ it is enough to compute something with the following structure



where G is in P and, at each step, the first input (pointer) increases one bit and the second input (recursion input) decreases one bit. At the root of the tree the pointer is ϵ , and at all leaves the recursion input is ϵ . Therefore, what we really have is



Intuitively this is clearly in PP. In order to prove it we consider PTMs with as many tapes as the arity of F (and G). We also assume that each component of the input is placed in one of the tapes (by the order that they show up), i.e. the n -th input is on the n -th tape, and that the machines are initialized with the heads scanning the right most non-empty cell (if there is one).

Let BRANCH be a TM with initial state q_B , which works according to two transition functions – Left and Right. Informally speaking, Left adds the bit “0” at the end of the first input (in the first tape), and deletes the last bit of the second input (in the second tape). The heads move one cell to the right and one cell to the left, respectively. Right proceeds in an analogous way, but adds the bit “1”. This can be done in constant time.

G is in P, so there exists a deterministic TM, M_G , which computes the boolean function G , let us say, in time bounded by t_G . We denote by q_G the initial state of M_G , and we assume that the output – 0 or 1 – is written in the last tape.

A PTM for F , M_F , can then be described as follows:

- (1) if the head of the second tape scans ϵ , go to (3);
otherwise, go to state q_B and (2);
- (2) run BRANCH and (1);
- (3) go to state q_G and (4);
- (4) run M_G .

We say that final configurations of M_F are accepting configurations if, and only if, the rightmost bit of the last tape is 1.

To determine the computing time of M_F notice that the only possibility of going into a loop is when an instruction calls a previous one, i.e. in (2). Moreover, notice that, inputting p, z, \bar{x} to the machine, (1) is called $|z|$ -times, and each loop uses constant time. Therefore, before reaching the instruction (3) the machine performs $c \cdot |z|$ steps, for some constant c . (3) is one step, and (4) uses, at most, t_G steps (notice that the inputs change along the process, but the sum of their lengths remains constant). Thus M_F runs in time bounded by $c \cdot |z| +_{\mathbb{N}} t_G(|p| +_{\mathbb{N}} |z| +_{\mathbb{N}} \sum_i |x_i|) +_{\mathbb{N}} 1$.

M_F accepts (ϵ, z, \bar{x}) if, and only if, more than half of the computations of M_F on (ϵ, z, \bar{x}) end in the ACCEPT final state. Noticing that M_F final configurations are M_G final configurations, we have that M_F accepts (ϵ, z, \bar{x}) if, and only if, more than half of $G(p, \epsilon, \bar{x})$ with $|p| = |z|$ end by 1. Abbreviate $0 \cdots 0$ and $1 \cdots 1$, of length $|z|$, by $0^{|z|}$ and $1^{|z|}$ respectively. Considering the lexicographic order, between $0^{|z|}$ and $1^{|z|}$ we have all 0 – 1 words with $|z|$ bits. There are $2^{|z|}$ different paths p of length $|z|$. So, M_F accepts (ϵ, z, \bar{x}) if, and only if², $\sum_{p=0^{|z|}}^{1^{|z|}} \text{last-bit of } G(p, \epsilon, \bar{x}) > \frac{2^{|z|}}{2}$, or equivalently, if and only if $\sum_{p=0^{|z|}}^{1^{|z|}} 2 \cdot \text{last-bit of } G(p, \epsilon, \bar{x}) > 2^{|z|}$. This is exactly what TR[⊕] tests: \oplus doubles the last bit of it inputs and add them; $+$ adds the inputs; and $\#$ (at the root of the tree) tests whether the global sum (i.e. $\sum_{p=0^{|z|}}^{1^{|z|}} 2 \cdot \text{last-bit of } G(p, \epsilon, \bar{x})$) is greater than $2^{|z|}$ – it returns 1 if YES, and 0 otherwise. ◀

² In numeric notation.

35:10 The Probabilistic Class PP

Let us now show that any boolean function in \mathbf{ST}_{PP} can be seen as one in the class PP.

► **Proposition 10.** $\mathcal{B}(\mathbf{ST}_{PP})$ is contained in PP.

Proof. It is enough to show that, for all $f \in \mathbf{ST}_{PP}$, the function F such that $F(\bar{x}, \bar{y})$ is 1 if $f(\bar{x}; \bar{y})$ ends by 1, and $F(\bar{x}, \bar{y}) = 0$ otherwise, is in PP. We prove this by induction on the definition of the function terms inside \mathbf{ST}_{PP} .

Whenever the scheme $\text{TR}[\boxplus]$ is not involved in the definition of f , one has that $f \in \mathbf{ST}_P$ and therefore the result is immediate. Thus, the relevant cases are the ones where the $\text{TR}[\boxplus]$ is used in the definition of the function term. There are two cases:

■ First of all, let f be defined by $\text{TR}[\boxplus]$ with base function $g \in \mathbf{ST}_P \subseteq \mathbf{ST}_{PP}$. Let $G \in \text{PP}$ be given by induction hypothesis. Consider

$$F(p, z, \bar{x}) = \begin{cases} G(p, \epsilon, \bar{x}) & \text{if } z = \epsilon \\ 0 & \text{if } p, z \neq \epsilon \\ f(\epsilon, z, \bar{x};) & \text{if } p = \epsilon \wedge z \neq \epsilon. \end{cases}$$

F results from easy case distinctions involving three functions: G and the constant function equal to 0 are in PP. Notice that for $z \neq \epsilon$, $f(\epsilon, z, \bar{x};) = \text{TR}[\boxplus](g)(\epsilon, z, \bar{x};) = \text{TR}[\boxplus](\text{last-bit of } g)(\epsilon, z, \bar{x};)$, where **last-bit of** w is $\mathbf{C}(w, 0, 0, 1;)$. g belongs to FPTIME , so **last-bit of** g is a boolean function in P. Thus, for $z \neq \epsilon$, $f(\epsilon, z, \bar{x};)$ is equal to a function $-\text{TR}[\boxplus](\text{last-bit of } g)(\epsilon, z, \bar{x};) -$ which is in PP due to Lemma 9. So, $F \in \text{PP}$. It remains to prove that

$$F(p, z, \bar{x}) = \begin{cases} 1 & \text{if } f(p, z, \bar{x};) \text{ ends by 1} \\ 0 & \text{otherwise.} \end{cases}$$

Let us further distinguish three sub-cases:

1. If $z = \epsilon$, then

$$F(p, \epsilon, \bar{x}) = G(p, \epsilon, \bar{x}) = \begin{cases} 1 & \text{if } g(p, \epsilon, \bar{x};) \text{ ends by 1} \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } f(p, \epsilon, \bar{x};) \text{ ends by 1} \\ 0 & \text{otherwise.} \end{cases}$$

2. If $p, z \neq \epsilon$, then $f(p, z, \bar{x})$ is the value returned to the root of a tree as described in Figure 2(b) of Section 3, because the function $\#$ is not involved (the pointer only increases along the recursion, therefore if it is not ϵ at the root, it is never ϵ). Thus, as explained in Section 3, the value returned to root of the tree is the binary representation of an even number. Therefore, $f(p, z, \bar{x})$ does not end by the bit 1. Consequently, in this case, $F(p, z, \bar{x}) = 0$.

3. If $p = \epsilon$ and $z \neq \epsilon$ then, noticing that $f(\epsilon, z, \bar{x};)$ is a single bit, we have that

$$F(\epsilon, z, \bar{x}) = f(\epsilon, z, \bar{x};) = \begin{cases} 1 & \text{if } f(p, \epsilon, \bar{x};) \text{ ends by 1} \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the function $F \in \text{PP}$ defined above, is 1 if f ends by 1, and it is 0 otherwise.

■ If f is defined by SC_P , let us say $f(\bar{x}; \bar{y}) = h(\bar{r}(\bar{x};); \bar{s}(\bar{x}; \bar{y}))$ with $\bar{r}, \bar{s} \in \mathbf{ST}_P$. By induction hypothesis for h , the function H such that $H(\bar{x}, \bar{y})$ is 1 if $h(\bar{x}; \bar{y})$ ends by 1, $H(\bar{x}, \bar{y}) = 0$ otherwise, is in PP. Let M_H be a PTM computing H in polynomial time p_H . \bar{r}, \bar{s} are in \mathbf{ST}_P , so let $M_{\bar{r}}$ and $M_{\bar{s}}$ be the correspondent deterministic Turing machines working in polynomial time. One may define the desired machine for F in the obvious way. First

running the deterministic polytime machines $M_{\bar{r}}$ and $M_{\bar{s}}$ in order to produce the input $(\bar{r}(\bar{x}); \bar{s}(\bar{x}; \bar{y}))$. Let us say that this is done in time dominated by $p_{\bar{r}, \bar{s}}(|\bar{x}|, |\bar{y}|)$. Second, running M_H on this input. The resulting machine is a PTM which computes F and works in time dominated by $p_{\bar{r}, \bar{s}}(|\bar{x}|, |\bar{y}|) +_{\mathbb{N}} p_H(|\bar{r}(\bar{x})|, |\bar{s}(\bar{x}; \bar{y})|)$. Evoking now the monotonicity of the polynomials and knowing that the length of \mathbf{ST}_P functions (i.e. FPTIME functions) is polynomial bounded – let us say $|\bar{r}(\bar{x})| \leq q_{\bar{r}}(|\bar{x}|)$ and $|\bar{s}(\bar{x}; \bar{y})| \leq q_{\bar{s}}(|\bar{x}|, |\bar{y}|)$ for some polynomials $q_{\bar{r}}$ and $q_{\bar{s}}$ – we have that the working time of M_F , on the input \bar{x}, \bar{y} , is bounded by the polynomial $p_{\bar{r}, \bar{s}}(|\bar{x}|, |\bar{y}|) +_{\mathbb{N}} p_H(q_{\bar{r}}(|\bar{x}|), q_{\bar{s}}(|\bar{x}|, |\bar{y}|))$.

This finishes the proof. \blacktriangleleft

4.3 Wrapping Up

From Proposition 8 and Proposition 10 one concludes that

► **Theorem 11.** \mathbf{ST}_{PP} characterizes PP (i.e. $PP = \mathcal{B}(\mathbf{ST}_{PP})$).

This establishes a purely recursion theoretic characterization of the probabilistic class of complexity PP by adding a specific form of tree-recursion to \mathbf{FPTIME} functions. One should notice that the step function of the tree-recursion makes use of the pointer p and the recursion variable z . The same happens in the characterization of $\mathbf{FPSPACE}$ given in [13], but it contrasts with the similar characterization of \mathbf{NP} given in [14]. For \mathbf{NP} the tree-recursion scheme is actually a tree-iteration scheme, i.e. pointers and recursion variable are used only at the bottom (and not all the way along the tree). There the pointers and recursion variable are not taken as inputs of the step function. It is, for instance, not known which class one obtains by restricting the tree-recursion of [13] to tree-iteration.

Whenever working with the tree-recursion scheme or with restricted forms of it, as it is the case here and in the papers mention above, one adopts \mathbb{W} (i.e. 0 – 1 words) as base algebra, instead of \mathbb{N} . The pointers and the tree-recursion scheme have a natural formulation over \mathbb{W} , but the present work can be rewritten in numeric notation. Actually, the seminal paper of Bellantoni and Cook for \mathbf{FPTIME} [2], which uses recursion on notation only, is in numeric notation.

A similar characterization of PP can be obtained working in a non-sorted context, by use of explicit bounds on the recursion on notation scheme. Such formulation of PP is based on Cobham’s characterization of \mathbf{FPTIME} [4] and uses the $\mathbf{TR}[\boxplus]$ scheme neglecting the “;”. There is no need of imposing explicit bounds to the non-sorted version of $\mathbf{TR}[\boxplus]$, because due to the fixed step function the scheme is implicitly polynomial bounded.

5 Related Work

The recursion-theoretic approach to implicit computational complexity has proved to be remarkably robust, with many different classes between logarithmic and polynomial space characterized by various forms of recursion schemes [2, 9, 10, 11, 12, 13, 14]. Tree recursion with pointers, in particular, is known to be capable of capturing classes larger than \mathbf{P} . But no probabilistic class of complexity was known to admit a recursion-theoretic characterization, so our result is certainly novel. With the given syntactic approach, one cannot expect to capture truly semantic classes like \mathbf{BPP} and \mathbf{ZPP} , but it may serve as a recursion-theoretic substratum for characterizations of these classes. In addition, it may provide as basis for proof-theoretic characterizations as provided for \mathbf{FPTIME} and related classes by Strahm [15] or for the polynomial hierarchy of functions in [8].

Other logical approaches to computational complexity have faced the challenge of characterizing probabilistic classes. As an example, a study of randomization and derandomization in descriptive complexity is due to Eickmeyer and Grohe [5], who were also able to characterize BPP in fixed-point logic with counting. The relationships between theories of bounded arithmetic and probabilistic complexity classes have been studied by Jerábek [7]. The present work complements these approaches by exploiting the framework of safe/tiered recursion [2, 9].

References

- 1 José L. Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*. Springer Publishing Company, Incorporated, 2nd edition, 2012.
- 2 Stephen Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the polytime functions. *Comput. Complex.*, 2:97–110, 1992.
- 3 Stephen Bellantoni and Isabel Oitavem. Separating NC along the delta axis. *Theor. Comput. Sci.*, 318(1-2):57–78, 2004.
- 4 Alan Cobham. The intrinsic computational difficulty of functions. In *Logic, Methodology and Philosophy of Science: Proceedings of the 1964 International Congress (Studies in Logic and the Foundations of Mathematics)*, pages 24–30. North-Holland Publishing, 1965.
- 5 Kord Eickmeyer and Martin Grohe. Randomisation and derandomisation in descriptive complexity theory. *Log. Methods Comput. Sci.*, 7(3), 2011.
- 6 John Gill. Computational complexity of probabilistic turing machines. *SIAM J. Comput.*, 6(4):675–695, 1977.
- 7 Emil Jerábek. Approximate counting in bounded arithmetic. *J. Symb. Log.*, 72(3):959–993, 2007.
- 8 Reinhard Kahle and Isabel Oitavem. Applicative theories for the polynomial hierarchy of time and its levels. *Annals of Pure and Applied Logic*, 164(6):663–675, 2013.
- 9 Daniel Leivant. Ramified recurrence and computational complexity I: word recurrence and polytime. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 320–343. Birkhäuser, 1995.
- 10 Daniel Leivant and Jean-Yves Marion. Ramified recurrence and computational complexity II: substitution and poly-space. In *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *LNCS*, pages 486–500. Springer, 1994.
- 11 Peter Møller Neergaard. A functional language for logarithmic space. In Wei-Ngan Chin, editor, *Programming Languages and Systems: Second Asian Symposium, APLAS 2004, Taipei, Taiwan, November 4–6, 2004. Proceedings*, volume 3302 of *LNCS*, pages 311–326. Springer, 2004.
- 12 Isabel Oitavem. Characterizing NC with tier 0 pointers. *Math. Log. Q.*, 50(1):9–17, 2004.
- 13 Isabel Oitavem. Characterizing PSPACE with pointers. *Math. Log. Q.*, 54(3):323–329, 2008.
- 14 Isabel Oitavem. A recursion-theoretic approach to NP. *Ann. Pure Appl. Log.*, 162(8):661–666, 2011.
- 15 Thomas Strahm. Theories with self-application and computational complexity. *Information and Computation*, 185:263–297, 2003.

Parallel Polynomial Permanent Mod Powers of 2 and Shortest Disjoint Cycles

Samir Datta ✉

Chennai Mathematical Institute, India

Kishlaya Jaiswal ✉

Chennai Mathematical Institute, India

Abstract

We present a parallel algorithm for permanent mod 2^k of a matrix of univariate integer polynomials. It places the problem in $\oplus L \subseteq NC^2$. This extends the techniques of Valiant [26], Braverman, Kulkarni and Roy [3] and Björklund and Husfeldt [2] and yields a (randomized) parallel algorithm for shortest two disjoint paths improving upon the recent (randomized) polynomial time algorithm [2].

We also recognize the disjoint paths problem as a special case of finding disjoint cycles, and present (randomized) parallel algorithms for finding a shortest cycle and shortest two disjoint cycles passing through any given fixed number of vertices or edges.

2012 ACM Subject Classification Theory of computation → Parallel algorithms

Keywords and phrases permanent mod powers of 2, parallel computation, graphs, shortest disjoint paths, shortest disjoint cycles

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.36

Related Version *Full Version:* <https://arxiv.org/abs/2106.00714>

Funding *Samir Datta:* Partially funded by a grant from Infosys foundation and SERB-MATRICES grant MTR/2017/000480.

Acknowledgements We would like to thank Eric Allender for a discussion regarding what was known about field operations in $\oplus L$. We would also like to thank Partha Mukhopadhyay for his comments on a preliminary version of the paper. We thank the anonymous referees for helping us improve the presentation of the paper.

1 Introduction

The problem of computing the determinant of a matrix has been a very well studied problem in the past, and several fast (both sequential and parallel) algorithms are known. On the contrary, Valiant in his seminal paper [26] showed that computing permanent of a matrix, an algebraic analogue of determinant, is hard. However modulo 2, determinant and permanent are equal and so building up on this, he presented an algorithm for computing permanent of an integer matrix modulo small powers of 2. The algorithm uses Gaussian elimination which is known to be highly sequential and so it is desirable to have a parallel algorithm. This was resolved by [3] who presented a $\oplus L \subseteq NC$ algorithm.

Moreover, NC algorithms for computing determinant of matrices over arbitrary commutative rings are also known, e.g. [18]. We would like to ask a similar question for the permanent. One natural extension would be to consider the ring of polynomials with integer coefficients. In this paper, we present an NC algorithm to compute permanent of matrices over integer polynomials modulo 2^k for any fixed k .

► **Theorem 1.** *Let $k \geq 1$ be fixed and A be an $n \times n$ matrix of integer polynomials, such that the degree of each entry is at most $\text{poly}(n)$. We can compute $\text{perm}(A) \pmod{2^k}$ in $\oplus L \subseteq NC^2$*



© Samir Datta and Kishlaya Jaiswal;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 36; pp. 36:1–36:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In the second half of the paper, we consider some applications of our parallel polynomial permanent algorithm. One direct consequence is that we are now able to parallelize the shortest two disjoint paths problem [2]. Furthermore, we generalize this problem by adding more constraints on the disjoint paths - that the paths should pass through any given set of edges. This can also be viewed as a problem of finding two disjoint cycles, for which we present a randomized parallel algorithm, using the techniques from [28] and [2].

► **Theorem 2.** *Let $k \geq 1$ be fixed and G be an undirected graph with k marked vertices. We can find shortest two disjoint cycles passing through the marked vertices in $\oplus\text{L}/\text{poly}$ (and RNC).*

Finally, we notice that a similar approach gives us an algorithm to compute Hafnians modulo 2^k of symmetric matrices of integers. Unfortunately, unlike the case of the permanent, we weren't able to extend this to a parallel algorithm. But nevertheless it gives a direct proof of the fact that counting number of perfect matchings modulo 2^k , in any general graph, is in P, as proved in [3].

1.1 Historical Survey

The problem of computing permanent of an integer matrix was first shown to be NP-hard by Valiant, where he also presented a $O(n^{4k-3})$ running time algorithm to compute permanent modulo 2^k . It was also shown that computing permanent modulo any odd prime still remains hard. Zanko [29] gave a proof for hardness of permanent under many-one reductions strengthening the result from the weaker Turing reductions used by Valiant. Later, Braverman, Kulkarni and Roy [3] presented a parallel $\oplus\text{SPACE}(k^2 \log n)$ algorithm for computing permanent modulo 2^k . Björklund and Husfeldt [2] presented a $d^3 n^{O(k)}$ time algorithm to compute permanent modulo 2^k of matrices over integer polynomials where the entries are of degree at most d .

Finding k disjoint paths in a graph has been a well studied problem in the past: given a graph (undirected/directed) and k pairs of terminals $(s_i, t_i)_{1 \leq i \leq k}$, find k pairwise vertex-disjoint paths P_i from s_i to t_i , if they exist.

When k is not fixed (and is part of input) then the problem is known to be NP-hard even for undirected planar graphs [16]. Linear time algorithms are known when further restricting directed planar graphs to the case: when all terminals lie on outer face [25], or when all the s_i -terminals lie on one common face while all the t_i -terminals lie on another common face [22]. If we further ask for paths with minimal total length in the latter problem, then [7] presented a $O(kn \log n)$ running time algorithm to achieve the same.

When k is fixed, the problem remains NP-hard for directed graphs, even for $k = 2$ [11], who had also given a poly time algorithm for the restricted case of directed acyclic graphs. In the restricted setting of directed planar graphs, [24] presented a $n^{O(k)}$ running time algorithm, which was further improved to a fixed parameter tractable algorithm by [4].

Shifting our focus to undirected graphs, the celebrated work of Robertson and Seymour [23] gave a $O(n^3)$ algorithm for finding k disjoint paths in an undirected graph, for any fixed k . [6] gave a parallel algorithm for class of planar graphs where all the terminals lie either on one or two faces. All this while, the question of finding *shortest* disjoint paths in general undirected graphs, remained open for many years until recently, Björklund and Husfeldt [2] gave a polynomial time algorithm for finding the shortest two disjoint paths. For general k , this problem still remains open. Björklund and Husfeldt also gave a parallel algorithm to count shortest two disjoint paths but only for cubic planar graphs [1].

1.2 Our Techniques

To compute permanent of a matrix A over integer polynomials, we closely follow the analysis of [3] but immediately hit an obstacle. They give a reduction from $\text{perm}(A) \pmod{4}$ to several computations of $\text{perm}(\cdot) \pmod{2}$, which crucially uses the fact that \mathbb{Z}_2 is a field. More precisely, when mimicking the proof, firstly it is required to find a non-trivial solution of $Av = 0$ with the property that at least one of the entries of this vector is invertible. This fails¹ over $\mathbb{Z}_2[x]$. Moreover, their algorithm also uses the fact that a non-singular matrix admits a LU decomposition iff all the leading principal minors are non-zero, which is known to hold in general only for matrices over fields.

Therefore, replacing \mathbb{Z}_{2^k} with $\mathbb{Z}_{2^k}[x]$ in their analysis doesn't work as $\mathbb{Z}_2[x]$ isn't a field while \mathbb{Z}_2 is. Furthermore, any finite field \mathbb{F} of characteristic 2 only corresponds to modulo 2 arithmetic. We need a way to extend the field structure so that it supports modulo 2^k arithmetic as well. If \mathbb{F} was realized as $\mathbb{Z}_2[x]/(p(x))$ where $p(x)$ is irreducible over \mathbb{Z}_2 then a possible candidate is the ring $\mathbb{Z}[x]/(2^k, p(x))$. Therefore, the appropriate algebraic structure to consider would be the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$.

Now we see that replacing \mathbb{Z} with \mathfrak{R} solves the above mentioned problems in the analysis, primarily because of the fact that $\mathfrak{R} \pmod{2}$ is a finite characteristic 2 field. With a slight bit of modification in the proof, we achieve that: given a matrix A over \mathfrak{R} , we can find $\text{perm}(A) \pmod{2^k}$ or in other words if A is a matrix over $\mathbb{Z}[x]$, we can compute $\text{perm}(A) \pmod{2^k, p(x)}$.

We are still not done because our aim was to compute $\text{perm}(A) \pmod{2^k}$ over $\mathbb{Z}[x]$. To achieve that, we choose $p(x)$ such that its degree is larger than the degree of polynomial $\text{perm}(A)$. This requires doing computations over a large field. Alternatively, we develop a new way of interpolation over \mathfrak{R} , which allows us to choose $p(x)$ such that its degree is of logarithmic order of degree $\text{perm}(A)$, but with a tradeoff of computing several (polynomially many) more permanents. We present this technique for its novelty.

Wahlström [28] addressed the question of finding a cycle passing through given vertices. We ask if we can also find shortest such cycle. And furthermore, can we also find shortest two disjoint cycles passing through these vertices? We combine techniques of [28] and [2] to answer the above questions, by reducing them to computing permanents modulo 2 of 2^{k-1} and modulo 4 of $2^{k-1} + 2^{k-2}$ matrices respectively. These matrices are adjacency matrix of what we refer to as *pattern* graphs. Notice that for $k = 2$ finding shortest two disjoint cycles corresponds to finding shortest two disjoint paths (by connecting each pair of terminals with a common vertex), and in this case our pattern graphs are exactly those presented in [2].

1.3 Organization of the Paper

In section 2, we first introduce the preliminaries and the notation that we shall be using throughout this paper. In the next section 3, we present proof of our main theorem 1 about computing permanent modulo 2^k , following which we also discuss the complexity of certain computations over the ring \mathfrak{R} which shows that our algorithm is in $\oplus\text{L}$. We also present an alternative proof for our main theorem in section 4 which uses new techniques. Then we present applications of our result that is finding shortest disjoint cycles, in section 5.

Finally, we discuss an example of our permanent algorithm and we show that using the same techniques as above we can also compute Hafnians and hence it gives an alternate proof of the already known result that counting perfect matchings modulo 2^k is in P. These are listed in the appendix for lack of space.

¹ Let $A = \begin{pmatrix} x & x+1 \\ x & x+1 \end{pmatrix}$ then there does not exist any null vector of the form $\begin{pmatrix} f \\ 1 \end{pmatrix}$ or $\begin{pmatrix} 1 \\ f \end{pmatrix}$ for any $f \in \mathbb{Z}_2[x]$

2 Preliminaries

We begin by stating the definition of the complexity class $\oplus\text{L}$.

- **Definition 3.** $\oplus\text{L}$ is the class of decision problems solvable by an NL machine such that
 - If the answer is “yes”, then the number of accepting paths is odd.
 - If the answer is “no”, then the number of accepting paths is even.

By a slight abuse of notation, we say a function class is in $\oplus\text{L}$ if each of its bits can be computed in $\oplus\text{L}$. We will have the occasion to use only constantly many bits because we are working mod 2^k .

Given an $n \times n$ matrix $A = (a_{ij})_{i,j \in [n]}$, determinant and permanent of A are defined as

$$\det(A) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^n a_{i\sigma(i)} \quad \text{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^n a_{i\sigma(i)}$$

The permanent of a matrix can be regarded as the weighted sum of cycle covers of an undirected graph. This gives a combinatorial interpretation to a seemingly pure algebraic quantity. We shall use this bridge to illustrate an application of our parallel polynomial permanent.

Let G be a weighted undirected graph (not necessarily loopless) with the associated weight function w .

- **Definition 4.** We say $C \subseteq V(G) \times V(G)$ is a cycle cover of G if
 - $(u, v) \in C \implies \{u, v\} \in E(G)$
 - C is a union of vertex-disjoint simple directed cycles in G
 - every vertex is incident to some directed edge in C

Note: loops are allowed as simple cycles in the above definition.

- **Definition 5.** For any cycle cover C we denote the weight of C by $\tilde{w}(C) = \prod_{e \in C} w(e)$

The above definition is well-defined because any directed edge (u, v) or (v, u) in our cycle cover correspond to the same edge $\{u, v\}$ in our underlying undirected graph. And hence both these directed edges get the same weight, that is $w((u, v)) = w((v, u)) = w(\{u, v\})$. In literature, such type of weight functions are commonly referred to as *symmetric* weight functions.

- **Definition 6.** Let $V(G) = [n]$ then we say $A = (a_{ij})_{i,j \in [n]}$ is the $(n \times n)$ adjacency matrix of G if

$$a_{ij} = \begin{cases} w(e) & \text{if } e = \{i, j\} \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

- **Observation 7.** $\text{perm}(A) = \sum \tilde{w}(C)$ where the sum is taken over all cycle covers C of G

3 Permanent over \mathfrak{R} Mod 2^k

To begin with, we fix some general notation. Let $p(x)$ be an irreducible polynomial over $\mathbb{Z}_2[x]$ such that $\deg(p(x))$ is at most $\text{poly}(n)$. Denote by \mathbb{F} the finite field of characteristic 2 which is realized as $\mathbb{Z}_2[x]/(p(x))$ and by $\mathfrak{R}_k = \mathbb{Z}[x]/(2^k, p(x))$. In particular $\mathfrak{R}_1 \cong \mathbb{F}$. Now as already discussed, we essentially replace \mathbb{Z} by \mathfrak{R} in the algorithm of [3]. We are ready to state the main theorem.

► **Theorem 8.** *Let $k \geq 1$ be fixed and $A \in \mathfrak{R}^{n \times n}$. We can compute $\text{perm}(A) \pmod{2^k}$ in $\oplus L$*

Proof is by induction on k . We start with the base case $k = 1$. Note that $\text{perm}(A) \equiv \det(A) \pmod{2}$. Using corollary 17 we can find $\text{perm}(A) \pmod{2}$ in $\oplus L$. Now suppose $k > 1$, we shall reduce it to computing several such determinants modulo 2, all of which can be computed in parallel. In doing so, we first illustrate an algorithm which is sequential and then we shall see how to parallelize it.

3.1 Sequential algorithm for computing permanent modulo 2^k

We present the algorithm from [3] for computing permanent modulo 2^k but translated within our framework. Let $A = (a_{ij})_{i,j \in [n]} \in \mathfrak{R}^{n \times n}$ be such that $\det(A) \equiv 0 \pmod{2}$. Therefore we can find a non-zero vector $v \in \mathbb{F}^n$ such that $A^T v = 0$ over \mathbb{F} . Assume without loss of generality $v_1 = 1$.

Let r_i denote the i^{th} row of A and define A' to be the matrix where the 1^{st} row in matrix A is replaced with $\sum_i v_i r_i$. Now if we expand the permanent along the first row then we get

$$\text{perm}(A') = \sum_{i=1}^n v_i \text{perm}(A[1 \leftarrow i]) = \text{perm}(A) + \sum_{i=2}^n v_i \text{perm}(A[1 \leftarrow i]) \quad (1)$$

where $A[i \leftarrow j]$ is the matrix A but with i^{th} row replaced with the j^{th} row. For $I, J \subseteq [n]$ denote by $A[\widehat{I}, \widehat{J}]$ the matrix obtained from A by deleting rows indexed by I and columns indexed by J . With this equation, modulo 2^k computation reduces to modulo 2^{k-1} computations of the minors as follows:

$$\text{perm}(A') = \sum_{j=1}^n \left(\sum_{i=1}^n v_i a_{ij} \right) \text{perm}(A[\widehat{\{1\}}, \widehat{\{j\}}])$$

Since $A^T v = 0 \pmod{2}$, we can write $\sum_i v_i a_{ij} = 2b_j \pmod{2^k}$ for some $b_j \in \mathfrak{R}_k$, therefore, we can re-write the above permanents as:

$$\text{perm}(A') \pmod{2^k} = 2 \left(\sum_{j=1}^n b_j \text{perm}(A[\widehat{\{1\}}, \widehat{\{j\}}]) \pmod{2^{k-1}} \right)$$

Similarly, expanding $\text{perm}(A[1 \leftarrow i])$ along the 1^{st} and i^{th} rows, we get the reduction:

$$\begin{aligned} \text{perm}(A[1 \leftarrow i]) &= \sum_{j \neq i} a_{ij} a_{il} \text{perm}(A[\widehat{\{1, i\}}, \widehat{\{j, l\}}]) \\ \text{perm}(A[1 \leftarrow i]) \pmod{2^k} &= 2 \left(\sum_{j < l} a_{ij} a_{il} \text{perm}(A[\widehat{\{1, i\}}, \widehat{\{j, l\}}]) \pmod{2^{k-1}} \right) \end{aligned}$$

Substituting these equations back in 1, we get

$$\begin{aligned} \text{perm}(A) \pmod{2^k} &= 2 \left(\sum_{j=1}^n b_j \text{perm}(A[\widehat{\{1\}}, \widehat{\{j\}}]) \pmod{2^{k-1}} \right) \\ &\quad - 2 \sum_{i=2}^n v_i \left(\sum_{\substack{j, l=1 \\ j < l}}^n a_{ij} a_{il} \text{perm}(A[\widehat{\{1, i\}}, \widehat{\{j, l\}}]) \pmod{2^{k-1}} \right) \end{aligned}$$

36:6 Parallel Polynomial Permanent

Since addition and multiplication in \mathfrak{R}_k is in $\oplus\mathbb{L}$ (see 14) we get that $\text{perm}(A) \pmod{2^k}$ $\oplus\mathbb{L}$ -reduces to $\text{perm}(\cdot) \pmod{2^{k-1}}$. Hence by induction, we can compute $\text{perm}(A) \pmod{2^k}$ in $\oplus\mathbb{L}$, provided that $\text{perm}(A) \equiv 0 \pmod{2}$.

Let us see how to drop this assumption. We expand the permanent of A along the i^{th} row, then

$$\text{perm}(A) = \sum_j a_{ij} \text{perm}(A[\widehat{\{i\}}, \widehat{\{j\}}])$$

If $\text{perm}(A) \not\equiv 0 \pmod{2}$, then $\exists i, j$ such that $\text{perm}(A[\widehat{\{i\}}, \widehat{\{j\}}]) \not\equiv 0 \pmod{2}$. We can find this pair (i, j) by running over all n^2 possibilities and evaluating $\text{perm}(A[\widehat{\{i\}}, \widehat{\{j\}}]) \pmod{2}$ using corollary 17, as already discussed above. Consider the matrix C where all entries are same as A except the $(i, j)^{\text{th}}$ entry which is replaced with $a_{ij} + y$. Then, we get $\text{perm}(C) = \text{perm}(A) + y \text{perm}(A[\widehat{\{i\}}, \widehat{\{j\}}])$. Notice that $\text{perm}(A) + y \text{perm}(A[\widehat{\{i\}}, \widehat{\{j\}}]) \equiv 0 \pmod{2}$ is a linear equation in y over the field \mathbb{F} and so there exists a unique $y \in \mathbb{F}$ which satisfies this equation, which is $y_0 = \text{perm}(A) \text{perm}(A[\widehat{\{i\}}, \widehat{\{j\}}])^{-1} \pmod{2}$ and can be computed using corollaries 14, 15 and 17. Setting $y = y_0$ we get $\text{perm}(C) \equiv 0 \pmod{2}$, so we can compute $\text{perm}(C) \pmod{2^k}$ and then compute $\text{perm}(A[\widehat{\{i\}}, \widehat{\{j\}}])$ recursively as $A[\widehat{\{i\}}, \widehat{\{j\}}]$ is a smaller $(n-1) \times (n-1)$ size matrix. Hence we obtain $\text{perm}(A) = \text{perm}(C) - y_0 \text{perm}(A[\widehat{\{i\}}, \widehat{\{j\}}]) \pmod{2^k}$. This yields a sequential algorithm for computing permanent modulo 2^k over \mathfrak{R} .

3.2 Parallel algorithm for computing permanent modulo 2^k

The bottleneck was finding i, j such that $A[\widehat{\{i\}}, \widehat{\{j\}}]$ is non-singular over \mathbb{F} . We fix this by again appealing to the fact that we are working over a field, and modifying A such that all leading principal minors are non-zero. This modification essentially derives from the following fact.

► **Theorem 9** ([14, Corollary 1]). *Let A be an invertible matrix over a field \mathbb{F} , then all leading principal minors are non-zero iff A admits an LU decomposition*

Every invertible matrix admits a PLU factorization [14] so let $A = PLU$. Denote by $Q = P^{-1}$, then $QA = LU$. Since Q is also a permutation matrix, we get that $\text{perm}(QA) = \text{perm}(A)$ (because permanent is invariant under row swaps). Therefore, it suffices to give a $\oplus\mathbb{L}$ algorithm to find Q so that we can replace A by QA which is an invertible matrix such that all leading principal minors are non-zero. Thus computing $\text{perm}(A) \pmod{2^k}$ reduces to the problem of computing (in parallel) permanent modulo 2^k of $n-1$ matrices with $\text{perm} \equiv 0 \pmod{2}$. This gives a $\oplus\mathbb{L}$ algorithm to compute permanent modulo 2^k over \mathfrak{R} .

To find Q , we closely follow [9]. For each $1 \leq i \leq n$, let A_i be the matrix formed from A by only taking the first i columns. Let A_i^j matrix obtained from A_i by only taking the first j rows. We construct a set $S_i \subseteq [n]$ inductively as follows:

- Base case: $l \in S_i$ if $\text{rank}(A_i^l) = 1$ and $\text{rank}(A_i^k) = 0$ for all $k < l$
- Include $j \in S_i$ iff $\text{rank}(A_i^j) = 1 + \text{rank}(A_i^{j-1})$

Since $\text{rank}(A_i) = i$, we get $|S_i| = i$. Furthermore note that $S_i \subset S_{i+1}$. So let $S_1 = \{s_1\}$ and for each $i \geq 2$, denote by $s_i \in S_i \setminus S_{i-1}$. Consider the following permutation $Q = (n, s_n) \dots (2, s_2)(1, s_1)$. Thus Q is our desired permutation, such that QA has all leading principal minors non-zero.

As a corollary, we immediately get our desired result.

► **Corollary 10** (Theorem 1 restated). *Given an $n \times n$ matrix $A = (a_{ij})_{i,j \in [n]}$ over $\mathbb{Z}[x]$ with $\deg(a_{ij})$ at most $\text{poly}(n)$, we can compute $\text{perm}(A) \pmod{2^k}$ in $\oplus\mathbf{L}$ for any fixed $k \geq 1$*

Proof. Let $N = n \max\{\deg(a_{ij})\} + 1$ and choose $l = \lceil \log_3(N/2) \rceil$. Consider $p(x) = x^{2 \cdot 3^l} + x^{3^l} + 1$ which is irreducible over $\mathbb{Z}_2[x]$ (see [27] Theorem 1.1.28)

Since $\deg(p(x)) \geq N > \deg(\text{perm}(A))$, using this $p(x)$ in above theorem, we get $\text{perm}(A) \pmod{2^k}$ for any fixed k . ◀

► **Remark.** See appendix A for a complete worked out example of the above algorithm.

3.3 Complexity Analysis

We discuss the complexity results for arithmetic operations over the ring \mathfrak{R}_k and matrix operations over the field \mathbb{F} , which were required in our above algorithm. To begin with, we state a well-known fact about integer polynomial matrix multiplication modulo 2. This shall form our basis for showing computations over \mathbb{F} in $\oplus\mathbf{L}$.

► **Lemma 11** (Folklore [5]). *Let $A_1, A_2, \dots, A_n \in \mathbb{Z}_2[x]^{n \times n}$ then the product $A_1 A_2 \dots A_n$ can be computed in $\oplus\mathbf{L}$*

To obtain an analogous result over \mathbb{F} we first perform multiplication over $\mathbb{Z}_2[x]$ and then divide all entries by $p(x)$, using the following polynomial division, as demonstrated by Hesse, Allender and Barrington in [12], to get that iterated matrix product over \mathbb{F} is in $\oplus\mathbf{L}$

► **Lemma 12** ([12, Corollary 6.5]). *Given $g(x), p(x) \in \mathbb{Z}[x]$ of degree at most $\text{poly}(n)$, we can compute $g(x) \pmod{p(x)}$ in $\text{DLOGTIME} - \text{uniform } \text{TC}^0 \subseteq \mathbf{L}$*

In particular, it follows that given an irreducible polynomial $p(x)$ (over $\mathbb{Z}_2[x]$), then for any $g(x) \in \mathbb{Z}[x]$ of degree at most $\text{poly}(n)$ we can find $g(x) \pmod{2^k, p(x)}$ in $\oplus\mathbf{L}$, for any fixed $k \geq 1$.

► **Corollary 13.** *Let $A_1, A_2, \dots, A_n \in \mathbb{F}^{n \times n}$ such that the degree of each entry is at most $\text{poly}(n)$ then the product $A_1 A_2 \dots A_n$ can be computed in $\oplus\mathbf{L}$*

Now we discuss arithmetic over \mathfrak{R}_k

► **Lemma 14.** *Let $k \geq 1$ be fixed then the following operations can be done in $\oplus\mathbf{L}$*

- *Multiplication : Given $a, b \in \mathfrak{R}_k$ compute ab*
- *Iterated Addition: Given $c_1, c_2, \dots, c_n \in \mathfrak{R}_k$ compute $\sum_i c_i$*

Proof. We use the fact that the arithmetic operations mentioned in the statement of lemma, but over \mathbb{Z}_{2^k} are in $\oplus\mathbf{L}$ (see for e.g. [12])

- Let $a(x), b(x) \in \mathfrak{R}_k$ and write $a(x) = \sum_{i=0}^D a_i x^i$ and $b(x) = \sum_{i=0}^{D'} b_i x^i$, then $a(x)b(x) = \sum_{i=0}^{D+D'} \left(\sum_{j=0}^i a_j b_{i-j} \right) x^i$. Finally, using lemma 12, divide $a(x)b(x)$ by $p(x)$ to obtain $ab \in \mathfrak{R}_k$
- Similarly, let $c_1(x), c_2(x), \dots, c_n(x) \in \mathfrak{R}_k$ and write $c_i(x) = \sum_{j=0}^{D_i} c_{ij} x^j$ for each $i \in [n]$, then $\sum_{i=1}^n c_i(x) = \sum_{j=0}^{\max\{D_i\}} \left(\sum_{i=1}^n c_{ij} \right) x^j$ where we assume $c_{ij} = 0$ if $j > D_i$. ◀

Our algorithm also requires computing inverse of non-zero elements. To compute inverse over \mathbb{F}^* we adopt the techniques from Fich and Tompa [8, 10]. Since $\mathbb{F} = \mathbb{Z}_2[x]/(p(x))$ with $N = \deg(p(x))$ which is at most $\text{poly}(n)$, then given $a \in \mathbb{F}^*$, we observe that $a^{-1} = a^{q-2}$ where $q = 2^N = |\mathbb{F}|$.

36:8 Parallel Polynomial Permanent

We interpret this equation over $\mathbb{Z}_2[x]$, that is we need to compute $a(x)^{q-2} \pmod{p(x)}$ over $\mathbb{Z}_2[x]$. First we show how to compute $a(x)^2 \pmod{p(x)}$. Construct the $N \times N$ matrix Q whose i^{th} row $(Q_{i,0}, Q_{i,1}, \dots, Q_{i,N-1})$ is defined as:

$$\sum_{j=0}^{N-1} Q_{i,j} x^j = x^{2i} \pmod{p(x)}$$

for each $0 \leq i \leq N-1$. Matrix Q can be computed in $\oplus L$ using the division lemma 12. Then the elements of the row vector $(a_0, a_1, \dots, a_{N-1})Q$ are the coefficients of $a(x)^2 \pmod{p(x)}$ as explained in section 3 of [10]. Furthermore, the coefficients of $a(x)^{2^k} \pmod{p(x)}$ are given by $(a_0, a_1, \dots, a_{N-1})Q^k$. From lemma 11 we get that $a(x)^{2^k} \pmod{p(x)}$ can be computed in $\oplus L$, for any k bounded by $\text{poly}(n)$. Therefore, writing $q-2 = (c_0, c_1, \dots, c_{N-1})$ in binary,

$$a(x)^{q-2} \pmod{p(x)} = \prod_{i=0}^{N-1} a(x)^{c_i 2^i} \pmod{p(x)}$$

can be computed in $\oplus L$, which gives us a^{-1} .

► **Corollary 15.** *Let $a \in \mathbb{F}$ then $a^{-1} \in \mathbb{F}$ can be computed in $\oplus L$*

Mahajan and Vinay [18] describe a way to reduce the computation of a determinant over a commutative ring to a semi-unbounded logarithmic depth circuit with addition and multiplication gates over the ring. In fact, the following is an easy consequence of their result:

► **Lemma 16 (Mahajan-Vinay [18]).** *Let $A \in R^{n \times n}$ be a matrix over a commutative ring. Then there exist $M \in R^{(2n^2) \times (2n^2)}$ and two vectors $a, b \in R^{2n^2}$ such that $\det(A) = a^T M b$. Moreover, each entry of the matrix M_{ij} and the vectors a, b is one of the entries $A_{i',j'}$ or a constant from $\{0, 1\}$ and the mapping ϕ where for every $(i, j) \in [2n^2] \times [2n^2]$, $\phi(i, j) \in A_{[n] \times [n]} \cup \{0, 1\}$ is computable in Logspace.*

Proof. (Sketch) In [18], given a matrix A they construct a graph H_A whose vertex set is $\{s, t_+, t_-\} \cup Q$ where $Q = \{[p, h, u, i] : p \in \{0, 1\}, h, u \in [n], i \in \{0, \dots, n-1\}\}$. Moreover, the edges are one of the following forms $(s, q), (q, q'), (q, t_+)$ and (q, t_-) where $q, q' \in Q$ and have weights $w(q, q')$ that each depend on a single entry of A or are one of the constants 0, 1. Moreover the mapping is very simple to describe. Let us focus on the induced subgraph $H_A[Q]$. Notice that $|Q| = 2n^3$ and each “layer” of $H_A[Q]$ is identical. In other words, $e_i = \langle [p, h, u, i], [p', h', u', i+1] \rangle$ is an edge in $H_A[Q]$ iff $e_j = \langle [p, h, u, j], [p', h', u', j+1] \rangle$ is an edge in $H_A[Q]$ and both have the same weights for every $i, j \in \{0, \dots, n-1\}$. Thus define the matrix M by putting $M_{[p,h,u],[p',h',u']}$ as the weight of any of the edges e_i .

Finally to define a, b : let $a_{[n \bmod 2, 1, 1]} = 1$ and $a_q = 0$ for all other q . $b_{[1, h, u]} = a_{uh}$ and $b_{[0, h, u]} = -a_{uh}$. The correctness of our Lemma then follows from the proof of Lemma 2 of [18]. ◀

Using above lemma 16, we reduce determinant over \mathbb{F} to matrix powering over \mathbb{F} , which can be computed in $\oplus L$ using corollary 13. Hence we get

► **Corollary 17.** *Let $A \in \mathbb{F}^{n \times n}$ then $\det(A)$ can be computed in $\oplus L$*

Now we demonstrate two results: computing rank and a null vector a matrix over \mathbb{F} in $\oplus L$. We use Mulmuley’s algorithm [19], which requires finding determinant over the ring $\mathbb{F}[y, t]$, which reduces to matrix powering over $\mathbb{F}[y, t]$ by the above result. We shall further reduce this to matrix powering over \mathbb{F} as follows: Let R be an arbitrary commutative ring. We associate with each polynomial $f(x) = \sum_{i=0}^{d-1} f_i x^i \in R[x]$ a $d \times d$ lower-triangular matrix

$$P(f) = \begin{bmatrix} f_0 & & & & \\ f_1 & f_0 & & & \\ f_2 & f_1 & f_0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ f_{d-1} & f_{d-2} & f_{d-3} & \dots & f_0 \end{bmatrix} \in R^{d \times d}$$

Suppose we have two polynomials $f(x)$ and $g(x)$ of degree d_1 and d_2 respectively. We can interpret them as degree $d_1 + d_2$ polynomials (with higher exponent coefficients as 0). Then we have that $P(f + g) = P(f) + P(g)$ and $P(fg) = P(f)P(g)$.

► **Theorem 18.** *Let R be any commutative ring and A_1, A_2, \dots, A_n be $n \times n$ matrices over $R[x]$ such that the degree of each entry is at most $\text{poly}(n)$. Denote by $\mathbf{A} = \prod A_i$. There exists $\text{poly}(n) \times \text{poly}(n)$ matrices B_1, B_2, \dots, B_n over R such that the coefficient of x^k in \mathbf{A}_{ij} is equal to $\mathbf{B}_{\psi(i,j,k)}$ where $\mathbf{B} = \prod B_i$ and ψ is logspace computable.*

In other words, iterated matrix multiplication over $R[x]$ is logspace reducible to iterated matrix multiplication over R .

Proof. Let $N = n \max_{i,j,k \in [n]} \{\deg((A_i)_{jk})\}$ where $(A_i)_{jk}$ denotes the $(j, k)^{\text{th}}$ entry of A_i . By our assumption, N is at most $\text{poly}(n)$. Now for each $1 \leq i \leq n$, compute the matrix $B_i \in R^{N \times N}$ obtained from A_i by replacing each polynomial $(A_i)_{jk}$ with the $N \times N$ matrix $P((A_i)_{jk})$. These matrices B_i can be computed in log space. Now the coefficient of x^k in \mathbf{A}_{ij} can be read from the entry $\mathbf{B}_{\psi(i,j,k)}$ where $\psi(i, j, k) = ((i-1)N + k + 1, (j-1)N + 1)$ is logspace computable. The correctness follows from our observation $P(fg) = P(f)P(g)$. ◀

► **Remark.** This gives us an alternate proof of the fact that iterated matrix multiplication over $\mathbb{Z}_2[x]$ is in $\oplus\mathbf{L}$, as it follows immediately from the definition of $\oplus\mathbf{L}$ that iterated matrix multiplication over \mathbb{Z}_2 is in $\oplus\mathbf{L}$.

► **Lemma 19** ([19]). *Let $A \in \mathbb{F}^{m \times n}$ then $\text{rank}(A)$ can be computed in $\oplus\mathbf{L}$*

Proof. We can assume that A is a square ($n \times n$) symmetric matrix because otherwise replace A with $\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$ which has rank twice that of A . Let Y be an $n \times n$ diagonal matrix with the $(i, i)^{\text{th}}$ entry as y^{i-1} . And let m be the smallest number such that t^m has a non-zero coefficient in the characteristic polynomial of YA , that is $\det(tI - YA)$. Then rank of $A = n - m$.

Suffices to show that $\det(tI - YA)$ can be computed in $\oplus\mathbf{L}$. Notice that $(tI - YA) \in \mathbb{F}[y, t]^{n \times n}$ and so $\det(tI - YA)$ is logspace reducible to matrix powering over $\mathbb{F}[y, t]$. Using the canonical isomorphism $\mathbb{F}[y, t] \cong \mathbb{F}[y][t]$, repeated application of theorem 18 logspace reduces it to matrix powering over \mathbb{F} . ◀

► **Observation 20.** *Let $A \in \mathbb{F}^{n \times n}$ be an invertible matrix then A^{-1} can be computed in $\oplus\mathbf{L}$*

This follows from the fact that computing A^{-1} involves computing the determinant of A and n^2 cofactors, that is determinants of n^2 matrices of size $(n-1) \times (n-1)$. Notice that this also requires inverting the determinant, an element of \mathbb{F}^* , which has been explained above.

► **Corollary 21.** *Let $A \in \mathbb{F}^{n \times n}$ then finding a non-trivial null vector (if it exists) is in $\oplus\mathbf{L}$*

36:10 Parallel Polynomial Permanent

Proof. Let $\text{rank}(A) = m$, then permute the rows and columns of A so that we can express $A = \begin{pmatrix} B & C \\ D & E \end{pmatrix}$ such that B is an invertible $m \times m$ matrix. Let $\begin{pmatrix} x \\ y \end{pmatrix}$ be a column vector where $x \in \mathbb{F}^m$ and $y \in \mathbb{F}^{n-m}$, such that

$$A \begin{pmatrix} x \\ y \end{pmatrix} = 0 \implies \begin{pmatrix} B & C \\ D & E \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 0$$

This reduces to the set of equations: $Bx + Cy = 0$ and $Dx + Ey = 0$. But the later is a redundant set of equations because $\begin{pmatrix} D & E \end{pmatrix}$ can be written in terms of $\begin{pmatrix} B & C \end{pmatrix}$. More precisely, there exists a matrix $V \in \mathbb{F}^{(n-m) \times m}$ such that $D = VB$ and $E = VC$ and so $Dx + Ey = VBx + VCy = V(Bx + Cy) = 0$. Therefore setting $x = \mathbf{1}$ and $y = -B^{-1}C\mathbf{1}$, gives us the desired null vector. So it suffices to give a $\oplus\text{L}$ algorithm to transform A to the form as specified above, which follows from [9]. Let A_i be the matrix formed from first i rows of A . We construct a set $S \subseteq [n]$ as follows:

- Base case: $i \in S$ if $\text{rank}(A_i) = 1$ and $\text{rank}(A_j) = 0$ for all $j < i$
- Include $k \in S$ iff $\text{rank}(A_k) = 1 + \text{rank}(A_{k-1})$

It follows that $|S| = m$ and let $S = \{i_1 < i_2 < \dots < i_m\}$ and P_r be the permutation matrix described by $(m, i_m) \dots (2, i_2)(1, i_1)$. Then $P_r A$ is the required matrix having first m rows as linearly independent. Next, consider the matrix $A' = (P_r A)^T$ and apply the above algorithm to get a permutation matrix P_c such that first m rows of $P_c A'$ are linearly independent. Then $P_r A P_c^T$ is the required matrix such that the leading principal m -minor is non-singular. ◀

4 Permanent via Interpolation

We now demonstrate another technique to compute permanent modulo 2^k , which doesn't resort to computations over exponentially sized fields. This proceeds by choosing small degree polynomial $p(x)$. The techniques developed in this section are new and hence interesting by themselves.

First we mention a result from [15] used to interpolate the coefficients of a polynomial.

► **Lemma 22** ([15, Lemma 3.1]). *Let \mathbb{F} be a finite, characteristic 2, field of order q .*

$$\sum_{a \in \mathbb{F}^*} a^m = \begin{cases} 0 & \text{if } q-1 \nmid m \\ 1 & \text{otherwise} \end{cases}$$

This dichotomy allows us to extract coefficients of any integer polynomial.

► **Lemma 23** ([15, Corollary 3.2]). *Let $f(x) = \sum_{i=0}^d c_i x^i$ be a polynomial with integer coefficients and $q > d+1$, then for any $0 \leq t \leq d$,*

$$\sum_{a \in \mathbb{F}^*} a^{q-1-t} f(a) = c_t \pmod{2}$$

But this gives us the coefficients modulo 2 only. How do we get coefficients modulo 2^k ?

The crucial observation here is that the above sum was computed over \mathbb{F} . So instead we do so over \mathfrak{R} by identifying a copy of $\mathbb{F}^* \hookrightarrow \mathfrak{R}$, and then we have

$$\sum_{a \in \mathbb{F}^*} a^m = \begin{cases} 2\alpha_m & \text{if } q-1 \nmid m \\ 2\beta_m + 1 & \text{otherwise} \end{cases}$$

where $\alpha_m, \beta_m \in \mathfrak{R}$.

Now we use repeated squaring method to obtain our desired modulo 2^k result.

► **Lemma 24.** $\forall m \geq 0, k \geq 1$

$$\sum_{a_1, \dots, a_{2^k-1} \in \mathbb{F}^*} (a_1 a_2 \cdots a_{2^k-1})^m = \left(\sum_{a \in \mathbb{F}^*} a^m \right)^{2^{k-1}} = \begin{cases} 0 \pmod{2^k} & \text{if } q-1 \nmid m \\ 1 \pmod{2^k} & \text{otherwise} \end{cases}$$

Note: We remind the reader that the computation here is done over \mathfrak{R}_k

Proof. Fix any $m \geq 0$. Clearly $(2\alpha_m)^{2^{k-1}} \equiv 0 \pmod{2^k}$.

Suffices to show $(2\beta_m + 1)^{2^{k-1}} \equiv 1 \pmod{2^k}$. This follows from induction on k . For $k = 1$ the result holds as stated above. Assume that for some $k \geq 1$, the result holds. Then we have $(2\beta_m + 1)^{2^{k-1}} = 2^k \gamma_{m,k} + 1$ where $\gamma_{m,k} \in \mathfrak{R}$

$$(2\beta_m + 1)^{2^k} = \left((2\beta_m + 1)^{2^{k-1}} \right)^2 = (2^k \gamma_{m,k} + 1)^2 = 1 \pmod{2^{k+1}} \quad \blacktriangleleft$$

Using this we can interpolate coefficients of an integer polynomial as follows:

$$\sum_{a_1, \dots, a_{2^k-1} \in \mathbb{F}^*} (a_1 \dots a_{2^k-1})^{q-1-t} f(a_1 \dots a_{2^k-1}) = c_t \pmod{2^k}$$

Finally let $A(x)$ be an $n \times n$ matrix of integer polynomials and the permanent polynomial be

$$\text{perm}(A(x)) = \sum_{i=0}^N c_i x^i$$

From the above lemma it follows that

$$\sum_{a_1, a_2, \dots \in \mathbb{F}^*} (a_1 a_2 \cdots)^{q-1-t} \text{perm}(A(a_1 a_2 \cdots)) = c_t \pmod{2^k}$$

provided that our field \mathbb{F} is of order at least $N + 2$. For this, fix $l = \lceil \frac{\log \log N}{\log 3} \rceil$ so that $2^{2 \cdot 3^l} > N + 1$. Hence the field obtained from the irreducible polynomial $p(x) = x^{2 \cdot 3^l} + x^{3^l} + 1$ ([27] Theorem 1.1.28) serves the purpose. It suffices to compute $|\mathbb{F}^*|^{2^{k-1}} = O(N^{2^{k-1}})$ many permanents over \mathfrak{R}_k to obtain all the coefficients c_t modulo 2^k , all of which can be computed in parallel. Hence, we can compute the permanent of A modulo 2^k over $\mathbb{Z}[x]$ in $\oplus \mathbb{L}$.

5 Shortest Disjoint Cycles

Now that we have a $\oplus \mathbb{L}$ algorithm to compute permanent mod 2^k for matrices over $\mathbb{Z}[x]$, we are all set to demonstrate a parallel algorithm for shortest two disjoint paths. But we notice that we can place this problem in a more general framework of shortest disjoint cycles. Let us first formally define these problems.

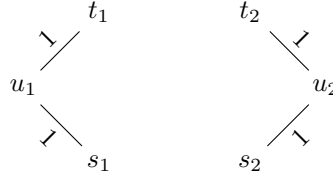
SDP(k): Given a weighted undirected graph with k pairs of marked vertices $\{(s_i, t_i) \mid 1 \leq i \leq k\}$, find the minimum of sum of weight of paths between each pair s_i and t_i such that all paths are pairwise disjoint.

SDC(l, k): Given a weighted undirected graph with k marked vertices, find the minimum of sum of weight of l cycles such that they pass through all the marked vertices and are pairwise disjoint and each cycle is incident to at least one of the marked vertices.

36:12 Parallel Polynomial Permanent

► **Note.** We only consider *non-trivial cycles* that is we don't consider self-loops in the above problem.

Given an instance of the $SDP(2)$ problem, join the pairs of vertices (s_1, t_1) and (s_2, t_2) with new vertices u_1 and u_2 respectively, as show in Figure 1. Notice that any two disjoint cycles passing through u_1 and u_2 give us two disjoint paths between (s_1, t_1) and (s_2, t_2) .



■ **Figure 1** Converting an instance of $SDP(2)$ to $SDC(2,2)$.

Similarly, connecting the k -pairs of vertices via another new vertex and edges of weight $x^0 = 1$, gives us a reduction from k disjoint paths to k disjoint cycles via k vertices. Since this reduction preserves the weight of the path/cycle, it is indeed a reduction from $SDP(k)$ to $SDC(k,k)$.

To apply the techniques of [2] to disjoint cycles problem, we instead consider the following variant $SDCE(l,k)$: Given a weighted undirected graph with k marked *edges*, find the minimum of sum of weight of l cycles such that they pass through all the marked edges and are pairwise disjoint.

It can be easily seen that for a fixed k there is a logspace reduction from $SDC(l,k)$ to $SDCE(l,k)$ as follows: Let $(G, \{v_1, v_2, \dots, v_k\})$ be an instance of $SDC(l,k)$. Assume without loss of generality that the marked vertices form an independent set, or otherwise split the edge into two by introducing a new vertex in the middle. For each i , choose a vertex u_i , a neighbour of v_i , such that $i \neq j \implies u_i \neq u_j$, we solve $(G, \{e_1, e_2, \dots, e_k\})$ an instance of $SDCE(l,k)$ where $e_i = \{u_i, v_i\}$ and output the smallest solution amongst all the instances of $SDCE$ thus created. Since for each i , $\deg(v_i) < n$, number of instances of $SDCE$ created are bounded by $O(n^k)$ all of which can be solved in parallel as k is fixed.

5.1 Pre-processing

Given a graph $G = (V, E, w)$ and k marked edges $\{e_i = \{s_i, t_i\}\}_{i \leq k}$, assign weight $x^{w(e)}$ to the edge e of G and add self loops (weight 1) on all vertices except $\{s_i, t_i\}_{i \leq k}$. Observe that all the non-zero terms appearing in the permanent of adjacency matrix correspond to a cycle cover in G . To force these k -edges in our cycle cover, we direct these edges in a certain way which we shall call as a *pattern*.

Formally, define a pattern P as an ordered pairing of terminals of given edges $\{s_i, t_i \mid 1 \leq i \leq k\}$. Furthermore, we view each undirected edge $\{u, v\}$ in G as two directed edges (u, v) and (v, u) with the same weight. For any pattern P , define a pattern graph G_P with the same vertex/edge set as of G but such that if $(u, v) \in P$ then all outgoing edges from u , except edge (u, v) , are deleted. We denote by A_P the adjacency matrix of G_P .

Now we shall show how to solve the $SDCE(1,k)$ problem for any $k \geq 1$. This algorithm essentially follows from the work of [28]. Next, we also present how to solve the $SDCE(2,k)$ problem for any $k \geq 2$. As far as we know, no algorithm (better than brute force) was known apriori to our work for $k \geq 3$.

5.2 Shortest Cycle

Let $\{e_i = \{s_i, t_i\}\}_{i \leq k}$ be given k -edges. For each binary sequence $b = (b_1, b_2, \dots, b_{k-1})$ of length $k - 1$, consider the following pattern P_b :

- $(s_1, t_1) \in P_b$
- $\forall 2 \leq i \leq k$, if $b_{i-1} = 0$ then $(s_i, t_i) \in P_b$ else $(t_i, s_i) \in P_b$

So $\{P_b\}_b$ is the collection of patterns with the orientation of e_1 fixed and all possible orientations of the other edges $\{e_i\}_{i \geq 2}$, as dictated by each binary sequence.

▷ **Claim 25.** Under the assumption that the shortest cycle is unique, the smallest exponent with non-zero coefficient in $f_1(x) \pmod{2}$ is the weight of unique shortest cycle passing through the given edges, where

$$f_1(x) = \sum_b \text{perm}(A_{P_b})$$

Proof. Let C be any cycle cover which consists of at least 2 non-trivial cycles. Consider the cycle in C which doesn't contain edge e_1 - there are two ways of orienting this cycle, namely clockwise and counter-clockwise. So this cycle cover contributes to $f_1(x)$ for at least two such b -sequences and so it vanishes modulo $f_1 \pmod{2}$.

Thus the only terms that survive in $f_1 \pmod{2}$ are the cycle covers which consist of one cycle passing through all the given edges and self-loops on the remaining vertices, and furthermore number of cycles of this weight must be odd.

Since the shortest weight cycle was unique by our assumption, we get the desired result. \triangleleft

To drop the assumption that a unique minimum weight solution exists, we instead assign modified weights $2nmw(e) + w'(e)$ where $n = |V(G)|$, $m = |E(G)|$, $w(e)$ is the given weight of edge e and $w'(e) \in \{0, 2, \dots, 2m - 1\}$ is chosen independently and uniformly at random for each edge e . Then isolation lemma [20] tells us that, with probability $\frac{1}{2}$, the minimum weight cycle is unique. Hence if the term x^j survives as the smallest exponent with non-zero coefficient term in $f_1(x) \pmod{2}$ then we get the weight of shortest cycle as $\lfloor j/2nm \rfloor$.

But this gives us only a randomized $\oplus L$ algorithm (that is RNC algorithm). To further show that a common poly weight scheme exists for all graphs of size n , we use the well-known result of [21] which immediately places this problem in $\oplus L/\text{poly}$. For completeness sake, we provide a proof of this fact in the appendix B.

5.3 Shortest Two Disjoint Cycles

We shall first prove the following stronger result:

► **Theorem 26.** *Given a set of k -edges $\{e_i\}_{i \leq k}$, we can find weight of the shortest two disjoint cycles passing through these edges such that e_1 and e_2 appear in different cycles in $\oplus L/\text{poly}$ (and RNC)*

Let $\{P_b\}_b$ be the patterns as defined above. Furthermore, for each binary sequence $c = (c_1, c_2, \dots, c_{k-2})$ of length $k - 2$, define pattern Q_c as

- $(s_1, s_2) \in Q_c$
- $(t_1, t_2) \in Q_c$
- $\forall 3 \leq i \leq k$, if $c_{i-2} = 0$ then $(s_i, t_i) \in Q_c$ else $(t_i, s_i) \in Q_c$

36:14 Parallel Polynomial Permanent

With a combination of these patterns, we can get our desired cycle covers. We claim that the non-zero terms appearing in

$$f_2(x) = \sum_b \text{perm}(A_{P_b}) - \sum_c \text{perm}(A_{Q_c})$$

correspond to cycle covers in G_{P_b} such that edges e_1 and e_2 appear in different cycles.

To prove our claim, we need to argue that the cycle covers of G_{P_b} in which e_1 and e_2 appear in the same cycle are exactly the cycle covers of G_{Q_c} . Let

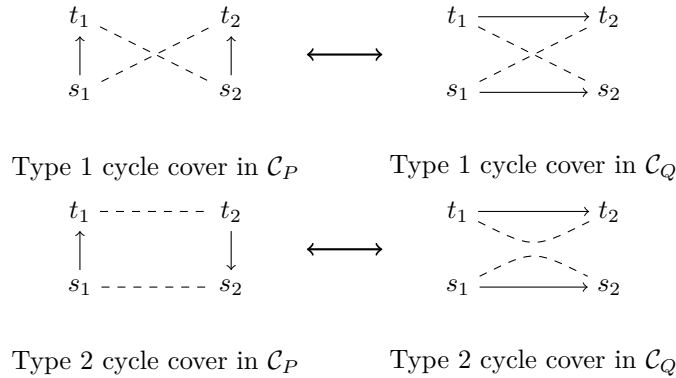
$$\mathcal{C}_Q = \bigsqcup_c \text{cycle covers of } G_{Q_c}$$

$$\mathcal{C}_P = \bigsqcup_b \text{cycle covers of } G_{P_b} \text{ such that edges } e_1 \text{ and } e_2 \text{ appear in the same cycle}$$

where each cycle cover is counted with repetitions in \mathcal{C}_P and \mathcal{C}_Q .

▷ **Claim 27.** There is a one-one correspondence between \mathcal{C}_P and \mathcal{C}_Q .

Proof. We define the mapping $\varphi : \mathcal{C}_P \rightarrow \mathcal{C}_Q$ as follows. Given a cycle cover in \mathcal{C}_P , remove the edges e_1 and e_2 and add edges (s_1, s_2) and (t_1, t_2) , refer to Figure 2 below.



■ **Figure 2** Bijection between \mathcal{C}_P and \mathcal{C}_Q .

To show that this is a well-defined mapping and indeed a bijection, we partition \mathcal{C}_P into type 1 and type 2 cycle covers, depending upon the orientation of the edge e_2 . Consider the cycle in which e_1 and e_2 appear together. Then if the edge e_2 is oriented as (s_2, t_2) then we call it type 1 cycle cover otherwise we call it a type 2 cycle cover.

Similarly, we partition \mathcal{C}_Q into type 1 and type 2 cycle covers. If the edges $\{s_1, s_2\}$ and $\{t_1, t_2\}$ appear in the same cycle then we call it a type 1 cycle cover otherwise we call it a type 2 cycle cover.

Fix a type 1 cycle cover of \mathcal{C}_P . Then it contains a cycle of the form

$$(s_1 \xrightarrow{e_1} t_1 \rightarrow P_1 \rightarrow s_2 \xrightarrow{e_2} t_2 \rightarrow P_2 \rightarrow s_1)$$

Applying φ to this cycle cover we get the cycle

$$(s_1 \xrightarrow{e_1} s_2 \rightarrow P_1^{\text{reverse}} \rightarrow t_1 \xrightarrow{e_2} t_2 \rightarrow P_2^{\text{reverse}} \rightarrow s_1)$$

and the other cycles remain intact. This constitutes a type 1 cycle cover in \mathcal{C}_Q

Similarly, consider a type 2 cycle cover of \mathcal{C}_P with the cycle

$$(s_1 \xrightarrow{e_1} t_1 \rightarrow P_1 \rightarrow t_2 \xrightarrow{e_2} s_2 \rightarrow P_2 \rightarrow s_1)$$

Applying φ to this cycle cover we get two cycles

$$(s_1 \xrightarrow{e_1} s_2 \rightarrow P_2^{\text{reverse}} \rightarrow s_1)$$

$$(t_1 \xrightarrow{e_2} t_2 \rightarrow P_1^{\text{reverse}} \rightarrow t_1)$$

and the other cycles remain intact. This constitutes a type 2 cycle cover in \mathcal{C}_Q .

Therefore, φ is a well-defined mapping and furthermore type i cycle covers of \mathcal{C}_P are mapped to type i cycle covers of \mathcal{C}_Q , $i \in \{1, 2\}$. Now consider $\psi : \mathcal{C}_Q \rightarrow \mathcal{C}_P$ defined as follows. Given a cycle cover in \mathcal{C}_Q , remove the edges (s_1, s_2) and (t_1, t_2) in the cycle and insert edges e_1 and e_2 with the orientation decided by the type. By an similar argument as above, we get that ψ is well-defined and clearly ψ is inverse of φ . \triangleleft

Now suppose C is a cycle cover of G such that edges e_1 and e_2 appear in different cycles. We have two cases:

Case 1: number of non-trivial cycles in C is more than 2. Consider any two non-trivial cycles in C such that e_1 is not incident on them. We can orient these two cycles in both clockwise and anti-clockwise direction and so we get that C is a cycle cover in G_{P_b} for at least 4 b -sequences. Hence, the term corresponding to C cancels out in $f_2 \pmod{4}$

Case 2: number of non-trivial cycles is exactly two. Then C is a cycle cover in G_{P_b} for exactly two b -sequences, that is the cycle passing through e_2 has two possible orientations whereas cycle passing through e_1 has a fixed orientation (as orientation of e_1 remains fixed in all G_{P_b}). Hence, the term corresponding to C appears with a coefficient of two in $f_2 \pmod{4}$. Therefore, the non-zero terms in $f_2 \pmod{4}$ correspond to only the cycle covers in which edges e_1 and e_2 appear in different cycles and number of non-trivial cycles is exactly 2. Assuming a unique shortest two disjoint cycle exists, it's weight can be obtained from the smallest exponent with a non-zero coefficient in $f_2 \pmod{4}$. Finally, to drop this assumption, we again assign random weights as done previously to ensure that the minimum weight solution is unique, with high probability. Furthermore, as in the previous case, we can again follow the proof of [21] to obtain a $\oplus\text{L/poly}$ algorithm. This completes the proof of Theorem 26.

► **Corollary 28** (Theorem 2 restated). *Given a set of k -edges $\{e_i\}_{i \leq k}$, we can find weight of the shortest two cycles passing through these edges in $\oplus\text{L/poly}$ (and RNC)*

Proof. For each pair of edges e_i and e_j , we can find weight of the shortest cycles separating them using the above algorithm. Hence taking the minimum over all pairs, we get our desired result. \blacktriangleleft

5.4 Constructing Cycles

We remark that under the assumption that the shortest cycle(s) are unique, we can recover these cycles C just from the knowledge of their weight $w(C)$. This follows the standard strategy of solving search via isolation as in [20]: for each edge $e \notin \{e_1, \dots, e_k\}$, delete the edge e and call the resulting graph G_e . Running our algorithm on $(G_e, \{e_1, \dots, e_k\})$, if the shortest cycle(s) weight is more than $w(C)$, then discard e otherwise add e to the set C , which gives us the required cycle(s). Doing this procedure in parallel for each edge e , it can be easily seen that we can recover C in $\oplus\text{L/poly}$ (or RNC).

6 Conclusion

We started by recognizing the appropriate algebraic structure \mathfrak{R} over which we can present a parallel algorithm to compute permanent modulo 2^k . Then we saw two techniques to get permanent over $\mathbb{Z}[x] \pmod{2^k}$ from $\mathfrak{R} \pmod{2^k}$. First method was to choose a large enough irreducible polynomial for our ring \mathfrak{R} . Another method was to interpolate over the ring \mathfrak{R} , which was an extension of the commonly known interpolation over finite fields.

Then we considered applications for parallel polynomial permanent. This includes a direct parallelization of the shortest two disjoint paths problem as given by [2]. Another direct application, although which required some modification, was finding shortest cycle passing through given vertices [28]. We further presented a common framework to view the above mentioned problems. This also aided us in further generalizing and obtaining a parallel algorithm to find shortest two disjoint cycles in any weighted undirected graph.

The more general question of computing permanent over arbitrary commutative rings of characteristic 2^k for $k \geq 2$ still remains open. On the other hand, using the framework we presented, we ask if it is possible to obtain shortest k disjoint cycles for $k \geq 3$ in RNC or even perhaps in randomized polynomial time?

References

- 1 Andreas Björklund and Thore Husfeldt. Counting shortest two disjoint paths in cubic planar graphs with an NC algorithm. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, volume 123 of *LIPIcs*, pages 19:1–19:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ISAAC.2018.19.
- 2 Andreas Björklund and Thore Husfeldt. Shortest two disjoint paths in polynomial time. *SIAM J. Comput.*, 48(6):1698–1710, 2019. doi:10.1137/18M1223034.
- 3 Mark Braverman, Raghav Kulkarni, and Sambuddha Roy. Space-efficient counting in graphs on surfaces. *Comput. Complex.*, 18(4):601–649, 2009. doi:10.1007/s00037-009-0266-4.
- 4 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The planar directed k -vertex-disjoint paths problem is fixed-parameter tractable. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 197–206. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.29.
- 5 Carsten Damm. Problems complete for parity. *Information Processing Letters*, 36(5):247–250, 1990. doi:10.1016/0020-0190(90)90150-V.
- 6 Samir Datta, Siddharth Iyer, Raghav Kulkarni, and Anish Mukherjee. Shortest k -disjoint paths via determinants. In Sumit Ganguly and Paritosh K. Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, volume 122 of *LIPIcs*, pages 19:1–19:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.19.
- 7 Éric Colin de Verdière and Alexander Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Trans. Algorithms*, 7(2):19:1–19:12, 2011. doi:10.1145/1921659.1921665.
- 8 W. Eberly. Very fast parallel matrix and polynomial arithmetic. In *25th Annual Symposium on Foundations of Computer Science, 1984.*, pages 21–30, 1984. doi:10.1109/SFCS.1984.715897.
- 9 W. Eberly. Efficient parallel independent subsets and matrix factorizations. In *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 204–211, 1991. doi:10.1109/SPDP.1991.218278.
- 10 Faith E. Fich and Martin Tompa. The parallel complexity of exponentiating polynomials over finite fields. *J. ACM*, 35(3):651–667, 1988. doi:10.1145/44483.44496.

- 11 Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980. doi:10.1016/0304-3975(80)90009-2.
- 12 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002. doi:10.1016/S0022-0000(02)00025-9.
- 13 Hiroshi Hirai and Hiroyuki Namba. Shortest $(a+b)$ -path packing via hafnian. *Algorithmica*, 80(8):2478–2491, 2018. doi:10.1007/s00453-017-0334-0.
- 14 Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, USA, 2nd edition, 2012.
- 15 Ce Jin, Nikhil Vyas, and Ryan Williams. Fast low-space algorithms for subset sum. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1757–1776. SIAM, 2021. doi:10.1137/1.9781611976465.106.
- 16 James F. Lynch. The equivalence of theorem proving and the interconnection problem. *SIGDA Newsl.*, 5(3):31–36, 1975. doi:10.1145/1061425.1061430.
- 17 Meena Mahajan, P. R. Subramanya, and V. Vinay. The combinatorial approach yields an nc algorithm for computing pfaffians. *Discret. Appl. Math.*, 143(1-3):1–16, 2004. doi:10.1016/j.dam.2003.12.001.
- 18 Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chic. J. Theor. Comput. Sci.*, 1997, 1997. URL: <http://cjtcs.cs.uchicago.edu/articles/1997/5/contents.html>.
- 19 Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Comb.*, 7(1):101–104, 1987. doi:10.1007/BF02579205.
- 20 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC '87*, page 345–354, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.383347.
- 21 Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM J. Comput.*, 29(4):1118–1131, 2000. doi:10.1137/S0097539798339041.
- 22 Heike Ripphausen-Lipa, Dorothea Wagner, and Karsten Weihe. Linear-time algorithms for disjoint two-face paths problems in planar graphs. *Int. J. Found. Comput. Sci.*, 7(2):95–110, 1996. doi:10.1142/S0129054196000087.
- 23 N. Robertson and P.D. Seymour. Graph minors .xiii. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.
- 24 Alexander Schrijver. Finding k disjoint paths in a directed planar graph. *SIAM J. Comput.*, 23(4):780–788, 1994. doi:10.1137/S0097539792224061.
- 25 Hitoshi Suzuki, Takehiro Akama, and Takao Nishizeki. Finding steiner forests in planar graphs. In David S. Johnson, editor, *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California, USA*, pages 444–453. SIAM, 1990. URL: <http://dl.acm.org/citation.cfm?id=320176.320230>.
- 26 Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. doi:10.1016/0304-3975(79)90044-6.
- 27 J.H. van Lint. *Introduction to Coding Theory*. Graduate Texts in Mathematics. Springer Berlin Heidelberg, 2013. URL: <https://books.google.co.in/books?id=6dbqCAAQBAJ>.
- 28 Magnus Wahlström. Abusing the tutte matrix: An algebraic instance compression for the k -set-cycle problem. In *STACS*, pages 341–352, 2013.
- 29 Viktória Zankó. $\#p$ -completeness via many-one reductions. *Int. J. Found. Comput. Sci.*, 2(1):77–82, 1991. doi:10.1142/S0129054191000066.

A

 Examples

Example 1

Let $A = \begin{pmatrix} 1 & x+1 & x+2 \\ x & x^2 & x^2+x \\ x^2 & 3 & x^2+3 \end{pmatrix}$, $p(x) = x^6 + x^3 + 1$ be the irreducible polynomial and we

want to evaluate $\text{perm}(A) \pmod{4}$ over the ring $\mathfrak{R} = \mathbb{Z}[x]/(p(x))$. First of all, a direct computation gives us $\text{perm}(A) = 2x^5 + 6x^4 + 2x^3 + 12x^2 + 12x$. Now we demonstrate the steps taken by our algorithm.

Step 1: We start by evaluating $\text{perm}(A) \pmod{2}$. We directly notice here that last column is the sum of first two columns and so $\det(A) = 0 \implies \text{perm}(A) \equiv 0 \pmod{2}$

Step 2: Since $\det(A) \equiv 0 \pmod{2}$, we solve the equation $A^T v = 0$ over \mathbb{F} by our method as

$$\text{follows: } \begin{pmatrix} 1 & x & x^2 \\ x+1 & x^2 & 1 \\ x & x^2+x & x^2+1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = 0$$

Since rank of the principal 2×2 submatrix is already 2, we set $v_3 = 1$ and solve the equation: $\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = - \begin{pmatrix} 1 & x \\ x+1 & x^2 \end{pmatrix}^{-1} \begin{pmatrix} x^2 \\ 1 \end{pmatrix}$ to get $v_1 = x^3 + 1$ and $v_2 = x^5 + x$.

Step 3: For each $j = 1, 2, 3$, we find b_j such that $\sum_i v_i a_{ij} = 2b_j \pmod{4}$

$$\begin{aligned} j = 1 : & \quad (x^3 + 1) + x(x^5 + x) + x^2 = 2x^2 \\ j = 2 : & \quad (x+1)(x^3 + 1) + x^2(x^5 + x) + 3 = 2x^3 \\ j = 3 : & \quad (x+2)(x^3 + 1) + (x^2 + x)(x^5 + x) + x^2 + 3 = 2x^3 + 2x^2 \end{aligned}$$

Step 4: We have the formula

$$\begin{aligned} \text{perm}(A) \pmod{4} &= 2 \left(\sum_{j=1}^3 b_j \text{perm}(A[\widehat{\{3\}}, \widehat{\{j\}}]) \pmod{2} \right) \\ &\quad - 2 \sum_{i=1}^2 v_i \left(\sum_{\substack{j,k=1 \\ j < k}}^3 a_{ij} a_{ik} \text{perm}(A[\widehat{\{3, i\}}, \widehat{\{j, k\}}]) \pmod{2} \right) \end{aligned}$$

Step 4.1:

$$\begin{aligned} \text{perm}(A[\widehat{\{3\}}, \widehat{\{1\}}]) &= \text{perm} \begin{pmatrix} x+1 & x+2 \\ x^2 & x^2+x \end{pmatrix} = x \pmod{2} \\ \text{perm}(A[\widehat{\{3\}}, \widehat{\{2\}}]) &= \text{perm} \begin{pmatrix} 1 & x+2 \\ x & x^2+x \end{pmatrix} = x \pmod{2} \\ \text{perm}(A[\widehat{\{3\}}, \widehat{\{3\}}]) &= \text{perm} \begin{pmatrix} 1 & x+1 \\ x & x^2 \end{pmatrix} = x \pmod{2} \\ \implies \sum_{j=1}^3 b_j \text{perm}(A[\widehat{\{3\}}, \widehat{\{j\}}]) &= ((x^3) + (x^4) + (x^4 + x^3)) = 0 \pmod{2} \end{aligned}$$

Step 4.2:

$$\begin{aligned}
& \sum_{\substack{j,k=1 \\ j < k}}^3 a_{1j}a_{1k} \text{perm}(A[\widehat{\{1,3\}}, \widehat{\{j,k\}}]) \\
&= (x+1)(x^2+x) + (x+2)x^2 + (x+1)(x+2)x = x^3 + x^2 + x \pmod{2} \\
& \sum_{\substack{j,k=1 \\ j < k}}^3 a_{2j}a_{2k} \text{perm}(A[\widehat{\{2,3\}}, \widehat{\{j,k\}}]) \\
&= x^3(x+2) + x(x^2+x)(x+1) + x^2(x^2+x) = x^4 + x^3 + x^2 \pmod{2} \\
& \sum_{i=1}^2 v_i \left(\sum_{\substack{j,k=1 \\ j < k}}^3 a_{ij}a_{ik} \text{perm}(A[\widehat{\{3,i\}}, \widehat{\{j,k\}}]) \pmod{2} \right) = x^5 + x^4 + x^3 \pmod{4}
\end{aligned}$$

Therefore, $\text{perm}(A) \pmod{4} = 2x^5 + 2x^4 + 2x^3$ which matches with our direct computation.

Example 2

Consider $A = \begin{pmatrix} 1 & x & x^2 \\ x & x^2 & 1 \\ 1 & x^2 & x \end{pmatrix}$, and so $\text{perm}(A) = x^5 + x^4 + x^2 + x$. Therefore, we now have $\det(A) \not\equiv 0 \pmod{2}$

Step 1: Find Q such that QA has all leading principal minors are non-zero. In this case, we

$$\text{will get } Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \implies QA = \begin{pmatrix} 1 & x & x^2 \\ 1 & x^2 & x \\ x & x^2 & 1 \end{pmatrix}$$

Step 2: Consider matrix C whose all entries are same as A except the last one which

is incremented by y , that is $C = \begin{pmatrix} 1 & x & x^2 \\ 1 & x^2 & x \\ x & x^2 & 1+y \end{pmatrix}$, then $\text{perm}(C) = \text{perm}(A) +$

$y \text{perm}(A[\widehat{\{3\}}, \widehat{\{3\}}])$. Again construct C' same as $A[\widehat{\{3\}}, \widehat{\{3\}}]$ but replace the last entry

incremented by y' , that is $C' = \begin{pmatrix} 1 & x \\ x & x^2 + y' \end{pmatrix} \implies \text{perm}(C') = \text{perm}(A[\widehat{\{3\}}, \widehat{\{3\}}]) +$

$y' \text{perm}(A[\widehat{\{2,3\}}, \widehat{\{2,3\}}])$. Written as one equation, we get

$$\text{perm}(A) = \text{perm}(C) - y(\text{perm}(C') - y'a_{11})$$

In this equation, both C, C' are matrices with $\det \equiv 0 \pmod{2}$ with the correct choice of y, y' , which were:

$$y_0 = \text{perm}(A) \text{perm}(A[\widehat{\{3\}}, \widehat{\{3\}}])^{-1} \pmod{2} = (x^5 + x^4 + x^2 + x)(x^4 + x^3 + x^2) = x^3 + 1$$

$$y'_0 = \text{perm}(A[\widehat{\{3\}}, \widehat{\{3\}}]) \text{perm}(A[\widehat{\{2,3\}}, \widehat{\{2,3\}}])^{-1} \pmod{2} = x^2 + x$$

So we can compute $\text{perm}(C)$ and $\text{perm}(C')$ by above method and substitute it into previous equation to get $\text{perm}(A) \pmod{4}$.

B Weights as a Polynomial Advice

Our proof is based on [21]. Call a weighted undirected graph (G, w) (w is the given weight function on edges) *min- k -unique*, if for any k marked edges on G , there exists unique shortest l disjoint cycles passing through these k edges. Our goal is to show for each $n > 0$ there exists a set of n^2 weight functions w_1, \dots, w_{n^2} such that given a graph G on n vertices, (G, w_i) is *min- k -unique* for some $i \in [1, n^2]$.

Given a graph G on n vertices and k marked edges e_1, \dots, e_k , let $\mathcal{F}(e_1, \dots, e_k)$ be the family of all l disjoint cycles passing through e_1, \dots, e_k . Using isolation lemma [20], if w is a random weight function, that is each edge is assigned a weight from $[1, 4n^{2k+2}]$ independently and uniformly at random, then probability that $\mathcal{F}(e_1, \dots, e_k)$ has a unique minimum weight element is at least $1 - 1/4n^{2k}$. Therefore, probability that (G, w) is not *min- k -unique* for a random weight function w is at most

$$\Pr[\exists e_1, \dots, e_k : \mathcal{F}(e_1, \dots, e_k) \text{ doesn't have a minimum weight element}] \leq \sum_{e_1, \dots, e_k} \frac{1}{4n^{2k}} \leq 1/4$$

Now we claim that there exists a set of n^2 weight functions $W = (w_1, \dots, w_{n^2})$ such that for any given graph G on n vertices, (G, w_i) is *min- k -unique* for some $1 \leq i \leq n^2$. We say W is *bad* if it doesn't meet this criteria and in particular W is *bad for G* , if none of (G, w_i) is *min- k -unique*. For a randomly chosen W , that is each w_i is chosen independently and uniformly at random, then

$$\begin{aligned} \Pr[W \text{ is bad for } G] &\leq \Pr[\forall i : (G, w_i) \text{ is not min-}k\text{-unique}] \leq \left(\frac{1}{4}\right)^{n^2} \\ \implies \Pr[W \text{ is bad}] &\leq \Pr[\exists G : W \text{ is bad for } G] \leq 2^{n^2} \left(\frac{1}{4}\right)^{n^2} < 1 \end{aligned}$$

Hence there exists some $W = (w_1, \dots, w_{n^2})$ which satisfies the above property and so W is the required poly advice. To complete the argument for $SDCE(1, k), SDCE(2, k) \in \oplus L$, we obtain the weight of shortest cycle(s), using each of the weight functions w_i and output the minimum amongst them.

C Hafnians and counting perfect matchings modulo 2^k

Similar to permanent and determinant, another pair of well-studied algebraic analogous functions on a matrix are hafnians and pfaffians. Let $A = (a_{ij})$ be a symmetric $2n \times 2n$ matrix over integers, hafnian is defined as

$$\text{hf}(A) = \frac{1}{2^n n!} \sum_{\sigma \in S_{2n}} \prod_{j=1}^n a_{\sigma(2j-1), \sigma(2j)} \quad (2)$$

Note that the diagonal entries of A don't contribute in the calculation of hafnians and hence we can assume them to be 0. Let $B = (b_{ij})$ be a skew-symmetric $2n \times 2n$ matrix, pfaffian is defined as

$$\text{pf}(B) = \frac{1}{2^n n!} \sum_{\sigma \in S_{2n}} \text{sgn}(\sigma) \prod_{j=1}^n b_{\sigma(2j-1), \sigma(2j)} \quad (3)$$

But notice that $\text{hf}(A) \equiv \text{pf}(A) \pmod{2}$. [17] have shown that $\text{pf}(A)$ can be computed in NC and hence as an immediate consequence we get that $\text{hf}(A) \pmod{2}$ can be computed in NC. We can reduce the computation of hafnian to several hafnians of smaller submatrices using the following lemma. Denote by $A[i, j]$ the matrix obtained from A by deleting rows i and j , columns i and j .

► **Lemma 29** ([13, Lemma 2.2]).

$$\text{hf}(A) = \sum_{j:j \neq i} a_{ij} \text{hf}(A[i, j])$$

$$\text{hf}(A) = a_{ij} \text{hf}(A[i, j]) + \sum_{pq:p, q \notin \{i, j\}, p \neq q} (a_{ip}a_{jq} + a_{iq}a_{jp}) \text{hf}(A[i, j, p, q])$$

Assume $\text{pf}(A) \equiv 0 \pmod{2}$, then $\det(A) \equiv 0 \pmod{2}$ and we can find a vector $v \in \mathbb{Z}_2^{2n}$ such that $Av = A^T v = 0 \pmod{2}$. Assume without loss of generality $v_1 = 1$.

Let r_i, c_i denote the i^{th} row and i^{th} column of A respectively.

- Construct A' by replacing first row with $\sum v_i r_i$ and then replacing first column with $\sum v_i c_i$
- Construct A_i by replacing first row with r_i and first column with c_i .

Then we check that

$$\begin{aligned} \text{hf}(A') &= \sum_{j>1} \left(\sum_{i \geq 1} v_i a_{ij} \right) \text{hf}(A[1, j]) \\ &= \sum_{i \geq 1} v_i \left(\sum_{j>1} a_{ij} \text{hf}(A[1, j]) \right) \\ &= \sum_{j>1} a_{1j} \text{hf}(A[1, j]) + \sum_{i>1} v_i \left(\sum_{j>1} a_{ij} \text{hf}(A[1, j]) \right) \\ &= \text{hf}(A) + \sum_{i>1} v_i \text{hf}(A_i) \end{aligned}$$

Computing $\text{hf}(A')$: since $A^T v = 0 \pmod{2} \implies \sum_{i \geq 1} v_i a_{ij} = 2b_j \pmod{2^k}$ for some $c_j \in \mathbb{Z}$ and hence

$$\begin{aligned} \text{hf}(A') &= \sum_{j>1} \left(\sum_{i \geq 1} v_i a_{ij} \right) \text{hf}(A[1, j]) \\ &= \sum_{j>1} 2b_j \text{hf}(A[1, j]) \\ \implies \text{hf}(A') \pmod{2^k} &= 2 \left(\sum_{j>1} b_j \text{hf}(A[1, j]) \pmod{2^{k-1}} \right) \end{aligned}$$

36:22 Parallel Polynomial Permanent

Computing $\text{hf}(A_i)$:

$$\begin{aligned} \text{hf}(A_i) &= a_{ii}\text{hf}(A[1, i]) + \sum_{pq:p, q \notin \{1, i\}, p \neq q} 2a_{ip}a_{iq}\text{hf}(A[1, i, p, q]) \\ \implies \text{hf}(A_i) \pmod{2^k} &= 2 \left(\sum_{pq:p, q \notin \{1, i\}, p \neq q} a_{ip}a_{iq}\text{hf}(A[1, i, p, q]) \pmod{2^{k-1}} \right) \end{aligned}$$

Thus, we can compute $\text{hf}(A) \pmod{2^k}$ provided $\text{pf}(A) \equiv 0 \pmod{2}$.

Now if $\text{pf}(A) \not\equiv 0 \pmod{2}$, then we can find $(i, j), i \neq j$ such that $\text{hf}(A[i, j]) \not\equiv 0 \pmod{2}$. Consider the matrix C where all entries are same as in A except a_{ij} is replaced with $a_{ij} + 1$, then we get $\text{hf}(C) = \text{hf}(A) + \text{hf}(A[i, j])$. Since $\text{hf}(C) \equiv 0 \pmod{2}$, we can compute $\text{hf}(C) \pmod{2^k}$ as described above and since $\text{hf}(A[i, j])$ is a $(n-2) \times (n-2)$ matrix, we compute it's hafnian recursively modulo 2^k . Therefore, we can compute $\text{hf}(A) = \text{hf}(C) - \text{hf}(A[i, j]) \pmod{2^k}$.

This gives us a P algorithm for computing hafnians modulo 2^k .

Counting perfect matchings modulo 2^k

Let G be an undirected graph and A_G denote the adjacency matrix of the graph G . If G has odd number of vertices, then clearly there aren't any perfect matchings. Otherwise it is straight-forward to see that number of perfect matchings in G is same as the value $\text{hf}(A_G)$. Hence the result follows.

On the Relative Power of Linear Algebraic Approximations of Graph Isomorphism

Anuj Dawar  

Department of Computer Science and Technology, University of Cambridge, UK

Danny Vagnozzi 

Department of Computer Science and Technology, University of Cambridge, UK

Abstract

We compare the capabilities of two approaches to approximating graph isomorphism using linear algebraic methods: the *invertible map tests* (introduced by Dawar and Holm) and proof systems with algebraic rules, namely *polynomial calculus*, *monomial calculus* and *Nullstellensatz calculus*. In the case of fields of characteristic zero, these variants are all essentially equivalent to the Weisfeiler-Leman algorithms. In positive characteristic we show that the distinguishing power of the monomial calculus is no greater than the invertible map method by simulating the former in a fixed-point logic with solvability operators. In turn, we show that the distinctions made by this logic can be implemented in the Nullstellensatz calculus.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Theory of computation → Proof complexity; Theory of computation → Complexity theory and logic

Keywords and phrases Graph isomorphism, proof complexity, invertible map tests

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.37

Related Version *Full Version:* <https://arxiv.org/abs/2103.16294>

Funding Research funded in part by EPSRC grant EP/S03238X/1.

Acknowledgements We want to thank Martin Grohe, Benedikt Pago and Gregory Wilsenach for useful discussions.

1 Introduction

The *graph isomorphism problem* consists in deciding whether there is an edge-preserving bijection between the vertex sets of two given graphs. Computationally, this problem is polynomial-time equivalent to finding the partition into orbits of the action of the automorphism group of a given graph on its vertex set. More generally, it is polynomial-time equivalent to computing the partition into orbits of the *induced* action of the automorphism group on the k^{th} power of the vertex set for any fixed k [20] (we shall refer to this partition as the *k-orbit partition* for a graph). The complexity of these problems is notoriously unresolved: while there are reasons to believe that they are not NP-complete, it is still an open problem as to whether they are in P. The best known upper bound to their computational time is quasi-polynomial, which follows from a breakthrough by Babai [2].

There has been a recent surge of interest in linear-algebraic approaches to the graph isomorphism problem (see for example [4, 12, 17, 19, 22]). In this paper, we consider two distinct methods for incorporating algorithms for solving linear systems into graph isomorphism solvers and compare them. The first is based on the use of algebraic proofs systems, such as the polynomial calculus and the second are generalizations of the Weisfeiler-Leman method, based on stability conditions and coherent algebras.



© Anuj Dawar and Danny Vagnozzi;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 37; pp. 37:1–37:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

For every $k \in \mathbb{N}$, the k -Weisfeiler-Leman algorithm is a generalization of naïve colour refinement, giving an approximation of the k -orbit partition. For a graph with vertex set V , each of these algorithms outputs in time $|V|^{O(k)}$ a canonical labelled partition of V^k satisfying a stability condition and respecting local isomorphism. Informally, they can be seen as forming a family of algorithms, each defining a notion of equivalence on both graphs and tuples of vertices of graphs. A result by Cai, Fürer and Immerman [7] shows how to construct graphs Γ_k of size $O(k)$ for which the k -Weisfeiler-Leman algorithm fails to produce the k -orbit partition. In the same paper, it is shown that the equivalence classes of the output partition of the k -Weisfeiler-Leman algorithm coincide with the equivalence classes of k -tuples of vertices distinguished by *counting logic* formulae with at most $k + 1$ variables. Thus, one deduces from the tight connection made by Immerman and Lander (see Theorem 2.3 in [9]), that the equivalence notions defined by the Weisfeiler-Leman family of algorithms delimit the expressive power of *fixed point logic with counting* (FPC). Intuitively, one such limitation is the expressibility of solvability of systems of linear equations over finite fields, since the above mentioned constructions by Cai, Fürer and Immerman are essentially graph encodings of systems of linear equations over \mathbb{Z}_2 [1]. This has therefore prompted research into families of algorithms graded by the natural numbers, whose notion of equivalence on tuples of vertices of graphs is conceptually a linear algebraic invariance over some field \mathbb{F} . One such family is that of the *invertible map tests* over a field \mathbb{F} , first defined in [12]. For any graph, the k^{th} algorithm of this family also produces a canonical labelled partition of k -tuples of its vertices, satisfying a stability condition and respecting local isomorphism, thus giving another notion of equivalence on both graphs and k -tuples of vertices thereof. For a fixed characteristic, the output of the k -invertible map tests is independent of the field; as such, \mathbb{F} will hereafter be taken to be a prime field without loss of generality. One can claim that the family of equivalences on tuples defined by the Weisfeiler-Leman algorithms and that defined by the invertible map tests over \mathbb{Q} *simulate* each other in the following sense. For every $k \in \mathbb{N}$, there is some $k' \in \mathbb{N}$ such that for any graph, any pair of k -tuples of its vertices distinguished by the k -Weisfeiler-Leman algorithm are distinguished by the k' -invertible map test over \mathbb{Q} . Conversely, for every $k \in \mathbb{N}$ there is a $k' \in \mathbb{N}$ such that for any graph, any pair of k -tuples of vertices distinguished by the k -invertible map test over \mathbb{Q} is distinguished by the k' -Weisfeiler-Leman algorithm. If the characteristic of \mathbb{F} is positive the former statement holds, but not the latter; indeed, it is shown in [18] and [10] how one can construct graphs $\Gamma_{k,p}$ for each $k \in \mathbb{N}$ and prime number p , for which the 3-invertible map test over \mathbb{Z}_p outputs the 3-orbit partition, but the output of the k -invertible map test over \mathbb{Z}_q with $q \neq p$ is strictly coarser than the k -orbit partition. A recent construction due to Lichter [19] suggests a way of getting graphs, for any value of k , on which the k -orbit partition cannot be obtained by the k -invertible map test over \mathbb{Z}_q for any q whatsoever.

Another approach to approximating the orbit partition is that of algebraic proof systems [3, 8]. These systems are the subject of very active study in the area of proof complexity. They have been studied specifically in the context of graph isomorphism in [4] and [17]. In particular, the proof systems studied are the *polynomial calculus* (PC), and the weaker *Nullstellensatz calculus* (NC) and *monomial calculus* (MC). Each of these gives, for a fixed field \mathbb{F} a set of rules \mathcal{R} dictating how new polynomials, with coefficients in \mathbb{F} , may be derived from an initial set of polynomials, which we shall refer to as *axioms*. In the context of graph isomorphism, we encode any graph Γ on a vertex set V as a set of axioms $\text{Ax}(\Gamma) \subset \mathbb{F}[\{x_{uv} | u, v \in V\}]$, i.e. a collection of polynomials over variables x_{uv} corresponding to potential edges in the graph. An \mathcal{R} -derivation of the polynomial $x_{u_1 v_1} x_{u_2 v_2} \dots x_{u_k v_k}$ can then be seen as a proof that the tuples $\vec{u}, \vec{v} \in V^k$ are distinguishable. Say that such a derivation has *degree* d if all

polynomials occurring in the derivation have degree at most d . For each of the calculi, fixing the degree d gives us a polynomial-time algorithm for checking the existence of a derivation and hence a polynomial-time approximation of the orbit partition for graphs. Again, we may restrict \mathbb{F} to a prime field without loss of generality.

The question we address in this paper is how the approximations of the orbit partition obtained by these algebraic proof systems compare with those we get from the invertible map test. In the case of fields of characteristic zero, the answer is quite clear, as both approaches yield algorithms that are (up to constant factors) equivalent to the Weisfeiler-Leman algorithms. This is shown for the invertible map tests in [13] and for the polynomial calculus in [17]. In the case of positive characteristic, we show (in Section 4) the definability of derivations of MC in FPS(p), an extension of fixed-point logic with quantifiers for the solvability of systems of linear equations over fields of characteristic p . This implies, in particular, that the approximation of the orbit partition obtained by MC in characteristic p is no finer than that obtained by the invertible map test in characteristic p .

► **Theorem 1.** *For any prime number p , $k \in \mathbb{N}$ and $\vec{u}, \vec{v} \in V^k$, there is a $k' \in \mathbb{N}$ such that if $x_{u_1 v_1} \dots x_{u_k v_k}$ has a degree k MC derivation over \mathbb{Z}_p from $\text{Ax}(\Gamma)$, then \vec{u} and \vec{v} are distinguished by the k' -invertible map test over \mathbb{Z}_p .*

In the other direction, we show that NC is able to simulate (as far as the graph isomorphism problem is concerned) PC in characteristic zero and at least MC in positive characteristic.

► **Theorem 2.**

1. *For any $k \in \mathbb{N}$ and $\vec{u}, \vec{v} \in V^k$, there is a $k' \in \mathbb{N}$ such that if $x_{u_1 v_1} \dots x_{u_k v_k}$ has a degree k PC derivation of over \mathbb{Q} from $\text{Ax}(\Gamma)$, then there is also a degree k' NC derivation over \mathbb{Q} from $\text{Ax}(\Gamma)$.*
2. *For any prime number p , $k \in \mathbb{N}$ and $\vec{u}, \vec{v} \in V^k$, there is a $k' \in \mathbb{N}$ such that if $x_{u_1 v_1} \dots x_{u_k v_k}$ has a degree k MC derivation over \mathbb{Z}_p from $\text{Ax}(\Gamma)$, then it also has a degree k' NC derivation over \mathbb{Z}_p from $\text{Ax}(\Gamma)$.*

From this, a strengthening of Theorem 6.3 in [4] also follows. Let V be the vertex set of $\Gamma_{k,p}$ as above.

► **Theorem 3.** *If $\vec{u}, \vec{v} \in V^3$ are not in the same equivalence class of the 3-orbit partition of $\Gamma_{k,p}$, then $x_{u_1 v_1} x_{u_2 v_2} x_{u_3 v_3}$ has a degree 3 NC derivation over \mathbb{Z}_p from $\text{Ax}(\Gamma_{k,p})$.*

Due to lack of space we omit the proofs of a number of results. These can be found in the extended version [14].

Notational conventions

All sets are finite unless stated otherwise. Given two sets V and I , a tuple in V^I is denoted by \vec{v} , and its i^{th} entry by v_i , for each $i \in I$. We use the notation $(v_i)_{i \in I}$ to denote the element of V^I with i^{th} element equal to v_i . We set $[k] = \{1, 2, \dots, k\} \subset \mathbb{N}$ and define $[k]^{(r)} = \{\vec{x} \in [k]^r \mid x_i \neq x_j \forall i, j \in [r], i \neq j\}$ for $r \leq k$. For any $\vec{v} \in V^k$, $\vec{u} \in V^r$, and $\vec{i} \in [k]^{(r)}$ we define $\vec{v} \langle \vec{i}, \vec{u} \rangle \in V^k$ to be the tuple with entries

$$(\vec{v} \langle \vec{i}, \vec{u} \rangle)_j = \begin{cases} u_{i_s} & \text{if } j = i_s \text{ for some } s \in [r] \\ v_j & \text{otherwise.} \end{cases}$$

In other words, $\vec{v} \langle \vec{i}, \vec{u} \rangle$ is the tuple obtained from \vec{v} by substituting the elements of \vec{u} in the positions specified by \vec{i} . Given two tuples $\vec{v} \in V^r$ and $\vec{w} \in V^s$, their *concatenation* is denoted by $\vec{v} \cdot \vec{w} \in V^{r+s}$. For a relation $R \subseteq V^2$ we define the *adjacency matrix* of R to be the $V \times V$ matrix whose (u, v) entry is 1 if $(u, v) \in R$ and 0 otherwise.

2 Preliminaries

2.1 Labelled partitions and refinement operators

A *labelled partition* of a set A is a map $\gamma : A \rightarrow X$, where X is a set of elements sometimes referred to as *colours*, and $\gamma(a)$ as the colour of a in γ . Denote the class of labelled partitions of A by $\mathcal{P}(A)$. For partitions \mathfrak{R} and \mathfrak{S} on A we write $\mathfrak{R} \preceq_A \mathfrak{S}$ and say that \mathfrak{S} *refines* \mathfrak{R} if, whenever $a, b \in A$ are in the same equivalence class of \mathfrak{S} , they are also in the same equivalence class of \mathfrak{R} . We extend the partial order \preceq_A to labelled partitions by writing $\gamma \preceq_A \rho$ to mean that the equivalence relation $\{(a, b) \mid \rho(a) = \rho(b)\}$ refines the relation $\{(a, b) \mid \gamma(a) = \gamma(b)\}$. Note that this does not require that the co-domains of γ and ρ are the same or indeed related in any way.

Define an action of $\text{Sym}(k)$ on V^k by setting \vec{v}^π to be the element of V^k with i^{th} entry $v_{\pi^{-1}(i)}$. $\gamma \in \mathcal{P}(V^k)$ is said to be *invariant* if $\gamma(\vec{u}) = \gamma(\vec{v})$ implies $\gamma(\vec{u}^\pi) = \gamma(\vec{v}^\pi)$ for all $\pi \in \text{Sym}(k)$ and $\vec{u}, \vec{v} \in V^k$.

For $t \in [k]$, the *t-projection* of γ is defined to be the labelled partition $\text{pr}_t \gamma \in \mathcal{P}(V^t)$ such that for all $\vec{u} \in V^t$

$$\text{pr}_t \gamma(\vec{u}) = \gamma(u_1, u_2, \dots, u_t, \dots, u_t).$$

For $\vec{v} \in V^k$, with t, k as above, we similarly define $\text{pr}_t \vec{v}$ to be the t -tuple (v_1, v_2, \dots, v_t) .

► **Definition 4** (Refinement operator). *A k -refinement operator is a map R which, for any set V , assigns to each $\gamma \in \mathcal{P}(V^k)$ a partition $R \circ \gamma \in \mathcal{P}(V^k)$ such that $\gamma \preceq R \circ \gamma$ and it is monotone; that is, $\gamma \preceq \rho \implies R \circ \gamma \preceq R \circ \rho$.*

We say that $\gamma \in \mathcal{P}(V^k)$ is *R-stable* if $R \circ \gamma = \gamma$. Given an $X \in \mathcal{P}(V^k)$, define a sequence of labelled partitions by $X^0 = X$ and $X^{i+1} = R \circ X^i$. Then, there is some s such that for all $i, i \geq s$ implies that X^i is *R-stable*. For the minimal such s we denote X^s by $[X]^R$.

In order to define the refinement operator leading to the invertible map test (in Section 3.1 below), we need the notion of character vectors. Let $\vec{v} \in V^k$ be a k -tuple of vertices, $\vec{i} \in [k]^{(2r)}$ a $2r$ -tuple of indices and $\gamma \in \mathcal{P}(V^k)$ a labelled partition of V^k . For a pair $\vec{x}, \vec{y} \in V^r$ of r -tuples of vertices, $\gamma(\vec{v} \langle \vec{i}, \vec{x} \cdot \vec{y} \rangle)$ is the colour of the tuple obtained by substituting \vec{x}, \vec{y} into \vec{v} in the positions specified by \vec{i} . For each $\sigma \in \text{Im}(\gamma)$, we define the $V^r \times V^r$ matrix χ_σ with 0/1 entries as the adjacency matrix of the relation $\{(\vec{x}, \vec{y}) \in (V^r)^2 \mid \gamma(\vec{v} \langle \vec{i}, \vec{x} \cdot \vec{y} \rangle) = \sigma\}$. The (\vec{i}, \vec{v}) -*character vector* of γ is then defined to be the tuple $\vec{\chi} = (\chi_\sigma)_{\sigma \in \text{Im}(\gamma)}$.

2.2 Extensions of first order and inflationary fixed point logics

We assume the reader has some familiarity with first-order (FO) and fixed-point logics (FP), and logical interpretations. Details can be found in [16]. Throughout the paper, for a logic \mathcal{L} , we denote by \mathcal{L}_k the class of all \mathcal{L} -formulae (over some pre-specified vocabulary) with at most k variables. We use \mathcal{C} to denote the *counting logics* as in [21]. Let \mathfrak{A} be a structure with universe V and fix $\vec{u}, \vec{v} \in V^k$. We say that some \mathcal{L}_k formula $\phi(\vec{z})$ *distinguishes* $(\mathfrak{A}, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}, \vec{z} \mapsto \vec{v})$ if either $\mathfrak{A} \models \phi(\vec{u})$ and $\mathfrak{A} \not\models \phi(\vec{v})$ or $\mathfrak{A} \not\models \phi(\vec{u})$ and $\mathfrak{A} \models \phi(\vec{v})$. For a logic \mathcal{L} we denote its extension via *solvability quantifiers* slv_p over a finite field of characteristic p by $\mathcal{L}+\text{S}(p)$. Let $\phi(\vec{x}, \vec{y}, \vec{z})$ be a formula of such a logic, where $\vec{x}, \vec{y}, \vec{z}$ are i, j, k -tuples of variables respectively. Then $\text{slv}_p(\vec{x}\vec{y}.\phi(\vec{x}, \vec{y}, \vec{z}))$ is also a $\mathcal{L}+\text{S}(p)$ formula. See [11, 17] for more about these quantifiers. The semantics of this quantifier is as follows. To each structure with universe V and k -tuple $\vec{v} \in V^k$, we associate the $V^i \times V^j$ matrix $S_\phi^{\vec{v}}$ over $\{0, 1\} \subseteq \mathbb{Z}_p$ with (\vec{r}, \vec{s}) -entry equal to 1 if, and only if, $\mathfrak{A} \models \phi(\vec{r}, \vec{s}, \vec{v})$. Then, $(\mathfrak{A}, \vec{z} \mapsto \vec{v}) \models \text{slv}_p(\vec{x}\vec{y}.\phi(\vec{x}, \vec{y}, \vec{z}))$ if, and only if, there is some $\vec{a} \in \mathbb{Z}_p^{V^j}$ such that $S_\phi^{\vec{v}} \vec{a} = \mathbf{1}$.

When \mathcal{L} is FP or FO, we denote $\mathcal{L}+\text{S}(p)$ by $\text{FPS}(p)$ or $\text{FOS}(p)$ respectively.

3 Refinement operators and proof systems with algebraic rules

We give an overview of the refinement operators and proof systems of interest.

3.1 The invertible map tests

The equivalence relations on tuples of vertices induced by the invertible map tests have been originally introduced in [12], under the guise of a pebble game with algebraic rules on a pair of graphs. Algorithmically, one can find these equivalence classes by computing a fixed point of the *invertible map operators* $\text{IM}_{k,r}^{\mathbb{F}}$, as defined in Sections 8 of [13]. Each of these is a k -refinement operator such that for $\gamma \in \mathcal{P}(V^k)$, the colour of $\vec{v} \in V^k$ in the partition $\text{IM}_{k,r}^{\mathbb{F}} \circ \gamma$ is given by a tuple whose entries are $\gamma(\vec{v})$ and the equivalence classes under matrix conjugation of the (\vec{i}, \vec{v}) -character vectors of γ , for all $\vec{i} \in [k]^{(2r)}$. Without going into the details, one can show that γ is $\text{IM}_{k,r}^{\mathbb{F}}$ -stable if for all $\vec{u}, \vec{v} \in V^k$ and $\vec{i} \in [k]^{(2r)}$

$$\gamma(\vec{u}) = \gamma(\vec{v}) \implies \exists M \in \text{GL}_{V^r} \text{ s.t. } \forall \sigma \in \text{Im}(\gamma)(\mathbb{F}), M\chi_{\sigma}M^{-1} = \xi_{\sigma}$$

where $(\chi_{\sigma})_{\sigma \in \text{Im}(\gamma)}$ and $(\xi_{\sigma})_{\sigma \in \text{Im}(\gamma)}$ are the (\vec{i}, \vec{v}) and (\vec{i}, \vec{u}) -character vectors of γ respectively.

For a graph Γ on V , let $\alpha_{k,\Gamma}$ be a canonical labelled partition of V^k into atomic types of Γ .¹ Define the k -refinement operator $\text{IM}_k^{\mathbb{F}}$ so that for $\gamma \in \mathcal{P}(V^k)$, the colour of $\vec{v} \in V^k$ in $\text{IM}_k^{\mathbb{F}} \circ \gamma$ is given by a tuple whose entries are the colours of \vec{v} in $\text{IM}_{k,r}^{\mathbb{F}} \circ \gamma$, for all $r \leq k/2$. Formally,

$$\text{IM}_k^{\mathbb{F}} \circ \gamma(\vec{v}) = (\text{IM}_{k,1}^{\mathbb{F}} \circ \gamma(\vec{v}), \text{IM}_{k,2}^{\mathbb{F}} \circ \gamma(\vec{v}), \dots, \text{IM}_{k,\lfloor k/2 \rfloor}^{\mathbb{F}} \circ \gamma(\vec{v})).$$

Then, the output of the k -invertible map test over \mathbb{F} is the labelled partition $[\alpha_{k,\Gamma}]^{\text{IM}_k^{\mathbb{F}}}$. It is explained in Proposition 4.6 in [13] how one can obtain this partition in time $|V|^{\mathcal{O}(k)}$ by iteratively applying $\text{IM}_k^{\mathbb{F}}$ to $\alpha_{k,\Gamma}$. Note that for a fixed characteristic, the choice of \mathbb{F} is irrelevant: indeed, if k -tuples $A, B \in \text{Mat}_V(\mathbb{F})^k$ are related by matrix conjugation over \mathbb{F} if, and only if, they are related by matrix conjugation over any field extension of \mathbb{F} [15]. Since the entries of the character vectors are 01-matrices, we may assume, without loss of generality, that \mathbb{F} is a prime field. Hereafter, we shall then indicate the operators $\text{IM}_{k,r}^{\mathbb{F}}$ and $\text{IM}_k^{\mathbb{F}}$ by $\text{IM}_{k,r}^c$ and IM_k^c respectively, where c is the characteristic of \mathbb{F} .

3.2 Counting logics operators

It is useful to express the partition of k -tuples into equivalence classes under finite variable counting logics as the fixed point of a refinement operator. For this purpose, the k -refinement operators $C_{k,r}$, for $r < k$, have been defined so that for any $\gamma \in \mathcal{P}(V^k)$, the colour of $\vec{v} \in V^k$ in $C_{k,r} \circ \gamma$ is given by a tuple whose entries are $\gamma(\vec{v})$ and the multisets of colours in γ of the tuples which can be obtained by substituting an r -tuple into \vec{v} (see Section 4 of [13]). In particular, γ is $C_{k,r}$ -stable if, and only if, for all $\vec{i} \in [k]^{(r)}$ and $\sigma \in \text{Im}(\gamma)$, the size of the set $\{\vec{x} \in V^r \mid \gamma(\vec{v}(\vec{i}, \vec{x})) = \sigma\}$ is independent of the choice of \vec{v} from within its equivalence class in γ . As such, for a graph Γ on V , $\vec{u}, \vec{v} \in V^k$ are in the same equivalence class of $[\alpha_{k,\Gamma}]^{C_{k,1}}$ if, and only if, there are no C_k formulae distinguishing $(\mathfrak{A}_{\Gamma}, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_{\Gamma}, \vec{z} \mapsto \vec{v})$. The combinatorial properties of $C_{k,r}$ -stable partitions can be used to show the relation between

¹ By *canonical* we mean *invariant under isomorphism*. That is, if Γ and Γ' are graphs on V and V' respectively and $\vec{u} \in V^k$ and $\vec{v} \in (V')^k$, then $\alpha_{k,\Gamma}(\vec{u}) = \alpha_{k,\Gamma'}(\vec{v})$ if, and only if, the mapping $u_i \rightarrow v_i$ is an isomorphism of the subgraphs induced by the vertices in \vec{u} and \vec{v} .

the distinguishing powers of finite variable fragments of counting logics and the invertible map tests. In short, the invertible map test over fields of characteristic zero is not more distinguishing than counting logic, but over fields of positive characteristic it is. To be precise, with Γ, \vec{u}, \vec{v} as above, the following holds:

For any field \mathbb{F} , if the k -invertible map test over \mathbb{F} does not distinguish \vec{u} and \vec{v} in Γ , then there are no \mathcal{C}_{k-1} -formulae distinguishing $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$.

If the k -invertible map test over \mathbb{Q} distinguishes \vec{u} from \vec{v} in Γ , then there is some \mathcal{C}_{2k-1} -formula distinguishing $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$.²

3.3 Solvability operators

In order to construct a refinement operator whose stable points reflect the properties of $\text{FOS}(p)$, we consider a weakened version of the invertible map operators, whose action on labelled partitions can be computed solely by solving systems of linear equations. To achieve this, we proceed by defining an equivalence relation \sim_{sol} on the character vectors, which can be seen as a relaxation of the conjugation relation \sim .

Let $\mathfrak{P}_V(\mathbb{F}) = \{A \in \text{Mat}_V(\mathbb{F}) \mid \sum_{w \in V} A_{wv} = \sum_{w \in V} A_{uw} = 1, \forall u, v \in V\}$. Or, equivalently, $\mathfrak{P}_V(\mathbb{F})$ is the set of matrices $A \in \text{Mat}_V(\mathbb{F})$ such that $\mathbf{1}_V$ is an eigenvector of both A and A^t , with corresponding eigenvalue 1. Set \mathbb{J}_V to be the $V \times V$ all-ones matrix and for a set I , let

$$\mathcal{X}_V^I(\mathbb{F}) = \{\vec{A} \in \text{Mat}_V(\mathbb{F})^I \mid \sum_{s \in I} A_s = \mathbb{J}_V \text{ and } \forall i \in I, \exists j, A_i^t = A_j\}.$$

Note that if γ is invariant and $\vec{i} \in [k]^{(2r)}$, then any (\vec{i}, \vec{v}) -character $\vec{\chi}$ of γ is an element of $\mathcal{X}_V^{\text{Im}(\gamma)}(\mathbb{F})$, since

$$\sum_{\sigma \in \text{Im}(\gamma)} \chi_\sigma = \mathbb{J}_{V^r} \quad (1)$$

and by invariance of γ , for any $\sigma \in \text{Im}(\gamma)$ there is some σ' such that $(\chi_\sigma)^t = \chi_{\sigma'}$. Define the relation \sim_{sol} on $\mathcal{X}_V^I(\mathbb{F})$ as follows: $\vec{A} \sim_{\text{sol}} \vec{B}$ if there is some $S \in \mathfrak{P}_V(\mathbb{F})$ such that $A_i S = S B_i$ for all $i \in I$.

► **Lemma 5.** \sim_{sol} is an equivalence relation on $\mathcal{X}_V^I(\mathbb{F})$.

Proof. Clearly, $\vec{A} \sim_{\text{sol}} \vec{A}$, since $\mathbb{I}_V \in \mathfrak{P}_V(\mathbb{F})$ and $A_i \mathbb{I}_V = \mathbb{I}_V A_i$ for all $i \in I$. Suppose $\vec{A} \sim_{\text{sol}} \vec{B}$. Let $S \in \mathfrak{P}_V(\mathbb{F})$ satisfy $A_i S = S B_i$ for all $i \in I$. Then $B_i^t S^t = S^t A_i^t$ and thus, from the definition of \mathcal{X}_V^I , $B_i S^t = S^t A_i$ for all $i \in I$. Since $S^t \in \mathfrak{P}_V(\mathbb{F})$, $\vec{B} \sim_{\text{sol}} \vec{A}$. Finally, suppose $\vec{A} \sim_{\text{sol}} \vec{B}$ and $\vec{B} \sim_{\text{sol}} \vec{C}$. Let $S, T \in \mathfrak{P}_V(\mathbb{F})$ satisfy $A_i S = S B_i$ and $B_i T = T C_i$ for all $i \in I$. Then $A_i S T = S B_i T = S T C_i$. Since S, T, S^t and T^t must all have $\mathbf{1}_{V^r}$ as eigenvector, with corresponding eigenvalue 1, so must ST and $(ST)^t$. Hence, $ST \in \mathfrak{P}_V(\mathbb{F})$ and $\vec{A} \sim_{\text{sol}} \vec{C}$. ◀

² This is a direct consequence of the following generalizations of Lemmata 7.1 and 7.3 in [13] respectively:
for all $k, r \in \mathbb{N}$ with $2r < k$,

1. The k -projection of a graph-like $\text{IM}_{k+r,r}^c$ -stable partition is $\text{C}_{k,r}$ -stable for any characteristic c .
2. The k -projection of a graph-like $\text{C}_{k+r,r}$ -stable partition is $\text{IM}_{k,r}^0$ -stable.

The authors prove it only for the case $r = 1$, but a similar argument holds for any $r \in \mathbb{N}$.

For $k, r \in \mathbb{N}$ with $2r \leq k$, a field \mathbb{F} , and an invariant $\gamma \in \mathcal{P}(V^k)$, we define the *solvability operators* $S_{k,r}^{\mathbb{F}}$ by setting $S_{k,r}^{\mathbb{F}} \circ \gamma$ to be the labelled partition for which the colour of $\vec{v} \in V^k$ is a tuple whose entries are $\gamma(\vec{v})$ and the equivalence classes under the relation \sim_{sol} of the (\vec{i}, \vec{v}) -character vectors of γ , for all $\vec{i} \in [k]^{(2r)}$. Formally:

$$S_{k,r}^{\mathbb{F}} \circ \gamma : \begin{array}{ccc} V^k & \rightarrow & \text{Im}(\gamma) \times (\mathcal{X}_{V^r}^{\text{Im}(\gamma)}(\mathbb{F}) / \sim_{\text{sol}})^{[k]^{(2r)}} \\ \vec{v} & \mapsto & (\gamma(\vec{v}), (\vec{\chi}_{\vec{i}})_{\vec{i} \in [k]^{(2r)}}), \end{array}$$

where $\vec{\chi}_{\vec{i}}$ is the (\vec{i}, \vec{v}) -character vector of γ .

As before, since the entries of the matrices in the character vector are all 0 and 1, we may restrict \mathbb{F} to being a prime field without loss of generality, and denote $S_{k,r}^{\mathbb{F}}$ by $S_{k,r}^c$, where $c = \text{char}(\mathbb{F})$. It is easy to show that $S_{k,r}^c$ is monotone on the class of invariant partitions of V^k and is thus a k -refinement operator when considered with this domain restriction.

For the remainder of this section, we assume that $\gamma \in \mathcal{P}(V^k)$ is invariant and that $\vec{\chi}$ and $\vec{\xi}$ are the (\vec{i}, \vec{v}) and (\vec{i}, \vec{u}) -character vectors of γ respectively, for some fixed $\vec{i} \in [k]^{(2r)}$. The following is a direct consequence of the definition of $S_{k,r}^c$.

► **Proposition 6.** *Let \mathbb{F} be the prime field of characteristic c . For all $k, r \in \mathbb{N}$, with $2r \leq k$, $S_{k,r}^c \circ \gamma(\vec{u}) = S_{k,r}^c \circ \gamma(\vec{v})$ if, and only if, $\gamma(\vec{u}) = \gamma(\vec{v})$ and for each $\vec{i} \in [k]^{(2r)}$ there exist some $M \in \mathfrak{P}_{V^r}(\mathbb{F})$ such that for all $\sigma \in \text{Im}(\gamma)$, $\chi_{\sigma} M = M \xi_{\sigma}$. In particular, γ is $S_{k,r}^c$ -stable if, and only if, for all $\vec{u}, \vec{v} \in V^k$ and $\vec{i} \in [k]^{(2r)}$*

$$\gamma(\vec{u}) = \gamma(\vec{v}) \implies \exists M \in \mathfrak{P}_{V^r}(\mathbb{F}) \text{ s.t. } \forall \sigma \in \text{Im}(\gamma), \chi_{\sigma} M = M \xi_{\sigma} \quad \forall \sigma \in \text{Im}(\gamma).$$

We now show some useful properties of the operators $S_{k,r}^c$.

► **Lemma 7.** *An $\text{IM}_{k,r}^c$ -stable partition is $S_{k,r}^c$ -stable.*

Proof. Let \mathbb{F} be the prime field of characteristic c . Suppose γ is $\text{IM}_{k,r}^c$ -stable and let $\gamma(\vec{u}) = \gamma(\vec{v})$. Then, for all $\vec{i} \in [k]^{(2r)}$, there is some $M \in GL_{V^r}(\mathbb{F})$ such that $M^{-1} \chi_{\sigma} M = \xi_{\sigma}$ for all $\sigma \in \text{Im}(\gamma)$. By equation 1, $M \mathbb{J}_{V^r} = \mathbb{J}_{V^r} M$ and thus, both M and M^t have $\mathbf{1}_{V^r}$ as eigenvector with corresponding non-zero eigenvalue $\lambda \in \mathbb{F}$. Since, $\frac{1}{\lambda} M \in \mathfrak{P}_{V^r}(\mathbb{F})$, the result follows. ◀

► **Lemma 8.** *If γ is $S_{k,r}^c$ -stable the following hold:*

1. *If $\gamma(\vec{u}) = \gamma(\vec{v})$, and γ is graph-like, then for all $\vec{i} \in [k]^{(2r)}$ and $\sigma \in \text{Im}(\gamma)$, $\{\vec{w} \in V^r \mid \gamma(\vec{u}(\vec{i}, \vec{w} \cdot \vec{w})) = \sigma\}$ is non-empty if, and only if, $\{\vec{w} \in V^r \mid \gamma(\vec{v}(\vec{i}, \vec{w} \cdot \vec{w})) = \sigma\}$ is non-empty.*
2. *If $c = 0$, then γ is $C_{k,2r}$ -stable.*

Note that if $c = 0$, the second statement implies the first (refer to Section 4 and 5 of [13] for more details on properties of $C_{k,r}$ -stability).

Proof. Suppose $\{\vec{w} \in V^r \mid \gamma(\vec{u}(\vec{i}, (\vec{w} \cdot \vec{w}))) = \sigma\}$ is non-empty. Since γ is graph-like, χ_{σ} must have all the non-zero entries on the diagonal. Hence, for any $M \in \mathfrak{P}_{V^r}(\mathbb{F})$, $\chi_{\sigma} M$ and, consequently $M \xi_{\sigma}$ must be non-zero. As such, $\{\vec{w} \in V^r \mid \gamma(\vec{v}(\vec{i}, (\vec{w} \cdot \vec{w}))) = \sigma\}$ is non-empty. The converse can be argued by symmetry, thus showing (1).

For (2), it suffices to show that: if $A, B \in \text{Mat}_V(\mathbb{F})$ are 01-matrices and $AM = MB$ for some $M \in \mathfrak{P}_V(\mathbb{F})$ then A and B have the same number of non-zero entries if $\text{char}(\mathbb{F}) = 0$. Indeed, note that if α and β are the number of non-zero entries of A and B respectively, then $\alpha \mathbb{J}_V = \mathbb{J}_V A \mathbb{J}_V$ and $\beta \mathbb{J}_V = \mathbb{J}_V B \mathbb{J}_V$. Since $M \mathbb{J}_V = \mathbb{J}_V M = \mathbb{J}_V$,

$$\alpha \mathbb{J}_V = \mathbb{J}_V A \mathbb{J}_V = \mathbb{J}_V A M \mathbb{J}_V = \mathbb{J}_V M B \mathbb{J}_V = \mathbb{J}_V B \mathbb{J}_V = \beta \mathbb{J}_V,$$

whence $\alpha = \beta$. ◀

In particular, if $r = 1$, statement (1) above implies that there is some $x \in V$ such that $\text{pr}_{k-1}\gamma(\text{pr}_{k-1}\vec{u}(i, x)) = \sigma$ if, and only if, there is some $y \in V$, such that $\text{pr}_{k-1}\gamma(\text{pr}_{k-1}\vec{v}(i, y)) = \sigma$. Furthermore, from (2) and Lemma 5.7 in [13], $\text{pr}_{k-1}\gamma$ is C_{k-1} stable.

► **Corollary 9.** *Let $\gamma = [\alpha_{k,\Gamma}]^{S_{k,1}^c}$ for some graph Γ on V . If $\text{pr}_{k-1}\gamma(\vec{u}) = \text{pr}_{k-1}\gamma(\vec{v})$, there are no first-order formulae distinguishing $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$. In addition, if $c = 0$, there are no C_{k-1} -formulae distinguishing $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$.*

Similarly to the invertible map operators, we define the k -refinement operator S_k^c so that for $\gamma \in \mathcal{P}(V^k)$, the colour of $\vec{v} \in V^k$ in $S_k^c \circ \gamma$ is given by a tuple whose entries are the colours of \vec{v} in $S_{k,r}^c \circ \gamma$, for all $r < k/2$; that is,

$$S_k^c \circ \gamma(\vec{v}) = (S_{k,1}^c \circ \gamma(\vec{v}), S_{k,2}^c \circ \gamma(\vec{v}), \dots, S_{k,\lfloor k/2 \rfloor}^c \circ \gamma(\vec{v})).$$

The next statement will be the crux of our main results.

► **Theorem 10.** *For any prime number p , if $\gamma = [\alpha_{k,\Gamma}]^{S_k^p}$ and $\gamma(\vec{u}) = \gamma(\vec{v})$, there are no $\text{FOS}_{k-1}(p)$ formulae distinguishing $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$.*

Proof. We proceed by induction on the structure of $\text{FOS}(p)$ formulae $\phi(\vec{z})$, where \vec{z} is a k -tuple of pairwise distinct variables. If $\phi(\vec{z})$ contains only atomic formulae, boolean connectives and first order quantifiers, the statement holds by Corollary 9. Assume that for some $\phi(\vec{z}) \in \text{FOS}_k(p)$, $\mathfrak{A}_\Gamma \models \phi(\vec{u}) \iff \mathfrak{A}_\Gamma \models \phi(\vec{v})$, and suppose $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v}) \models \text{slv}_p[\vec{x}\vec{y}.\phi(\vec{z}(\vec{i}, \vec{x} \cdot \vec{y}))]$, where $\vec{i} \in [k]^{(2r)}$ and \vec{x}, \vec{y} are r -tuples of distinct variables (distinct from variables in the tuple \vec{z}). Let $S^{\vec{v}}$ be the adjacency matrix of the relation $\{(\vec{a}, \vec{b}) \in V^r \times V^r \mid (\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v}) \models \phi(\vec{v}(\vec{i}, \vec{a} \cdot \vec{b}))\}$, and similarly define $S^{\vec{u}}$. Then, there is some $\vec{a} \in \mathbb{Z}_p^{V^r}$ such that $S^{\vec{v}}\vec{a} = \mathbf{1}_{V^r}$. By the induction hypothesis, $S^{\vec{u}} = \sum_{\sigma \in I} \chi_\sigma$ and $S^{\vec{v}} = \sum_{\sigma \in I} \xi_\sigma$ for some $I \subseteq \text{Im}(\gamma)$. Since there exists $M \in \mathfrak{P}_{V^r}(\mathbb{Z}_p)$ such that $\chi_\sigma M = M\xi_\sigma$ for all $\sigma \in \text{Im}(\gamma)$, we have

$$S^{\vec{v}}\vec{a} = \mathbf{1}_{V^r} \implies MS^{\vec{v}}\vec{a} = \mathbf{1}_{V^r} \implies S^{\vec{u}}(M\vec{a}) = \mathbf{1}_{V^r},$$

from which it follows that $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u}) \models \text{slv}_p[\vec{x}\vec{y}.\phi(\vec{z}(\vec{i}, \vec{x} \cdot \vec{y}))]$. Using a symmetric argument, we conclude that $\text{slv}_p[\vec{x}\vec{y}.\phi(\vec{z}(\vec{i}, \vec{x} \cdot \vec{y}))]$ does not distinguish $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$ and the result follows by induction. ◀

3.4 Polynomial, monomial and Nullstellensatz calculi

The idea behind *polynomial calculus* (PC), *monomial calculus* (MC) and *Nullstellensatz calculus* (NC) is that of encoding Boolean formulae as multivariate polynomials and concluding that they are inconsistent if the polynomials do not have a common root. The PC inference rules for a set of axioms $A \subset \mathbb{F}[x_1, \dots, x_n]$ are as follows:

1. \bar{f} for all $f \in A$.
2. *Multiplication rule:* $\frac{f}{xf}$ for all derived polynomials f , and variables $x \in \{x_1, \dots, x_n\}$.
3. *Linearity rule:* $\frac{f,g}{\lambda f + \mu g}$ for all derived polynomials f, g and $\lambda, \mu \in \mathbb{F}$.

The inference rules for MC are obtained by restricting f in the multiplication rule to be an axiom times a monomial or a monomial. By further restricting f to be an axiom times a monomial, one obtains the NC inference rules. The *degree* of a PC (or MC or NC) derivation is the maximum degree of all polynomials involved in the derivation, and a PC (or MC or NC, respectively) *refutation* of A is a derivation of 1 from A using PC (or MC or NC, respectively) rules. We are really interested in roots where the variables are assigned 0/1-values, so as to encode the Boolean framework. To enforce this, we assume that the axioms always includes the polynomials $x^2 - x$ for all $x \in \{x_1, \dots, x_n\}$. We may therefore restrict our focus to multilinear polynomials exclusively.

We denote by PC_k the proof system using the same inference rules as PC with the added constraint that all derivations must have degree at most k , and we use the same convention for MC_k and NC_k . Though these proof systems with bounded degree are not complete, their refutations can be decided in polynomial time in the number of variables.

For a graph Γ on V we define $\text{Ax}(\Gamma) \subseteq \mathbb{F}[\{x_{uv} \mid u, v \in V\}]$ to be the set of axioms containing the following polynomials:

1. $\sum_{u \in V} x_{uv} - 1$ for all $v \in V$.
2. $\sum_{u \in V} x_{vu} - 1$ for all $v \in V$.
3. $x_{uv}x_{u'v'}$ if the map $u \mapsto u', v \mapsto v'$ is not a local isomorphism in Γ .
4. $x_{uv}^2 - x_{uv}$ for all $u, v \in V$.

For $\vec{u}, \vec{v} \in V^k$, we further define $\text{Ax}(\Gamma_{\vec{u} \rightarrow \vec{v}})$ to contain the above plus $x_{v_i u_i} - 1$ for all $i \in [k]$.

When considering these axiom we assume, without loss of generality, that \mathbb{F} is a prime field. Let $\equiv_{\text{PC}_k}^c$ be the relation on V^k , where $\vec{u} \equiv_{\text{PC}_k}^c \vec{v}$ if there is no degree k PC refutation of $\text{Ax}(\Gamma_{\vec{u} \rightarrow \vec{v}})$ over the prime field of characteristic c , and similarly define $\equiv_{\text{MC}_k}^c$ and $\equiv_{\text{NC}_k}^c$.

► **Lemma 11.** $\equiv_{\text{PC}_k}^c, \equiv_{\text{MC}_k}^c$ and $\equiv_{\text{NC}_k}^c$ are equivalence relations on V^k .

Proof. Clearly, $\vec{u} \equiv_{\text{PC}_k}^c \vec{u}$. Indeed, the polynomials in the ideal generated by $\text{Ax}(\Gamma_{\vec{u} \rightarrow \vec{u}})$ have the common root $x_{rs} = \delta_{rs}$, where δ_{rs} is the Kronecker delta. Hence, the ideal generated by $\text{Ax}(\Gamma_{\vec{u} \rightarrow \vec{u}})$ is non-trivial. The set $\text{Ax}(\Gamma)$ is invariant under the transformation $x_{rs} \rightarrow x_{sr}$ for all $r, s \in V$. Thus, $\vec{u} \equiv_{\text{PC}_k}^c \vec{v} \implies \vec{v} \equiv_{\text{PC}_k}^c \vec{u}$.

Suppose $\vec{u} \equiv_{\text{PC}_k}^c \vec{v}$ and $\vec{v} \equiv_{\text{PC}_k}^c \vec{w}$. Let π be the map $v_i \rightarrow w_i$ for $i \in [k]$. Note that π is well defined, for if $v_i = v_j$ for some $i \neq j$ and $\pi(v_i) \neq \pi(v_j)$, then $x_{v_i w_i} x_{v_j w_j} \in \text{Ax}(\Gamma)$ - a contradiction. For $A \subseteq V^2$, define A^π as follows:

$$A^\pi = \begin{cases} \{(r, \pi(s)) \mid (r, s) \in A\} & \text{if for all } (r, s) \in A \text{ there is some } j \text{ such that } s = v_j; \\ \{(r, \pi^{-1}(s)) \mid (r, s) \in A\} & \text{if for all } (r, s) \in A \text{ there is some } j \text{ such that } s = w_j; \text{ and} \\ A & \text{otherwise.} \end{cases}$$

For a multilinear polynomial $f = \sum a_A X_A$, let $f^\pi = \sum a_A X_{A^\pi}$. We show by induction on the PC inference rules that there is a PC_k derivation of p if, and only if, there is PC_k derivation of f^π . Indeed, if f is in $\text{Ax}(\Gamma)$, then so is f^π . Suppose there is a PC_k derivation of f, g, f^π and g^π . Then there is a PC_k derivation of $(\lambda f + \mu g)^\pi = \lambda f^\pi + \mu g^\pi$. Finally, suppose the degree of f is less than k , and suppose, without loss of generality that $f = X_A$ for some $A \subseteq V^2$. Then, there is a PC_k derivation of $X_{A \cup \{(r,s)\}}$ for any $r, s \in V$. By checking case by case, it follows that there is a PC_k derivation of $(X_{A \cup \{(r,s)\}})^\pi$. Since $(A^\pi)^\pi = A$ and hence, $(f^\pi)^\pi = f$, there is a PC_k derivation of f if, and only if, there is a PC_k derivation of f^π . In particular, since there is no PC_k derivation of $X_{\{(v_i, w_i) \mid i \in [k]\}}$, there is no derivation of $(X_{\{(u_i, v_i) \mid i \in [k]\}})^\pi = X_{\{(u_i, w_i) \mid i \in [k]\}}$. Whence, $\vec{u} \equiv_{\text{PC}_k}^c \vec{w}$. ◀

It is easy to see that the relation $\equiv_{\text{PC}_k}^c$ refines $\equiv_{\text{MC}_k}^c$ which, in turn, refines $\equiv_{\text{NC}_k}^c$, since a NC_k refutation is a MC_k refutation which is also a PC_k refutation. More precisely:

► **Lemma 12.** For any graph on V and $\vec{u}, \vec{v} \in V^k$, $\vec{u} \equiv_{\text{PC}_k}^c \vec{v} \implies \vec{u} \equiv_{\text{MC}_k}^c \vec{v} \implies \vec{u} \equiv_{\text{NC}_k}^c \vec{v}$.

For $c = 0$, Grohe et al. have characterized these relations in terms of counting logics:

Let Γ be a graph on V and $\vec{u}, \vec{v} \in V^k$. Then $\vec{u} \equiv_{\text{MC}_k}^0 \vec{v}$ if, and only if, no \mathcal{C}_k formula distinguishes $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$ (Theorem 4.4 in [4]). Furthermore, if $\vec{u} \not\equiv_{\text{PC}_k}^0 \vec{v}$, there a $k' = O(k)$, such that some $\mathcal{C}_{k'}$ formula distinguishes $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$ (Theorem 6.6 in [17]).

Our main results attempt to give a similar characterization for $c > 0$ in terms of logics with solvability quantifiers.

4 Definability of monomial calculus refutations over finite fields

At the core of the proof of Theorem 1 is the definability of monomial calculus refutations in $\text{FPS}(p)$. More precisely, the main objective of this section is to prove the following statement (we will explain what we mean by structural encoding in Section 4.1).

► **Lemma 13.** *Let \mathfrak{A} be a structural encoding of a finite set of polynomials P of degree at most d , over a finite field \mathbb{F} of positive characteristic p . For any $k \in \mathbb{N}$ there is a $\text{FPS}(p)$ formula $\phi_{d,k}$ such that $\mathfrak{A} \models \phi_{d,k}$ if, and only if, there is an MC_k refutation of P over \mathbb{F} .³*

For the sake of argument, we assume that $\mathbb{F} = \mathbb{Z}_p$. We first recall how to express the solvability of linear equations with coefficients other than 0 and 1.

4.1 Defining solvability of linear equations over finite fields

For each prime number p , let LIN_p be a relational vocabulary with the following symbols:

1. A binary relational symbol \mathbf{A}_q for each $q \in \mathbb{Z}_p$.
2. A unary relational symbol \mathbf{b}_q for each $q \in \mathbb{Z}_p$.

Let $A \in \text{Mat}_{E \times V}(\mathbb{Z}_p)$ and $\vec{b} \in \mathbb{Z}_p^E$. A LIN_p -structure \mathfrak{A} with universe $V \cup E$ (V for variables and E for equations) is a structural encoding of the system of linear equations $A\vec{x} = \vec{b}$ if, for all $e \in E, v \in V$, $\mathfrak{A} \models \mathbf{A}_q(e, v)$ if $A_{ev} = q$ and $\mathfrak{A} \models \mathbf{b}_q(e)$ if $b_e = q$.

Recall Lemma 4.1 in [11].

► **Lemma 14.** *There is a quantifier free interpretation \mathcal{I} of LIN_p into LIN_p such that if \mathfrak{A} encodes the system of linear equations $A\vec{x} = \vec{b}$, then:*

1. $\mathcal{I}(\mathfrak{A})$ encodes a system of linear equations $A'\vec{y} = \mathbf{1}$, where $\mathbf{1}$ is the all 1s vector of appropriate length and A' is a 01-matrix.
2. $A'\vec{y} = \mathbf{1}$ has a solution if, and only if, $A\vec{x} = \vec{b}$ has a solution.

Thus, $\mathcal{I}(\mathfrak{A}) \models \text{slv}_p(xy.\mathbf{A}_1(x, y))$ if, and only if, $A\vec{x} = \vec{b}$ has a solution and hence, there is a $\text{FOS}(p)$ formula Φ such that $\mathfrak{A} \models \Phi$ if, and only if, the system encoded by \mathfrak{A} has a solution.

4.2 Idea of proof of Lemma 13

Deciding whether a set of axioms has a monomial calculus refutations of a given degree can be understood as the following procedure. The input is a finite set of multilinear polynomials P from the ring $\mathbb{F}[x_1, \dots, x_r]$, and the output is **REFUTE** or **NOREFUTE**. We denote the multilinear monomial $x_{a_1}x_{a_2} \dots x_{a_r}$ by X_A where $A = \{a_1, a_2, \dots, a_r\}$, so that for a polynomial f , $X_A f = x_{a_1}x_{a_2} \dots x_{a_r} f$. In this form, X_\emptyset denotes 1.

INPUT: $P \subset \mathbb{F}[x_1, \dots, x_r]$

OUTPUT: **REFUTE** or **NOREFUTE**.

Initialize $\mathcal{S} = \{X_A f \mid f \in P, \deg(X_A f) \leq k\}$.

while $\text{span}_{\mathbb{F}} \mathcal{S}$ has changed since last round or $1 \notin \text{span}_{\mathbb{F}} \mathcal{S}$ **do**

$\mathcal{M} \leftarrow \{A \subseteq [r] \mid |A| < k, X_A \in \text{span}_{\mathbb{F}} \mathcal{S}\}$.

$\mathcal{S} \leftarrow \mathcal{S} \cup \{X_B \mid |B| \leq k, \exists A \in \mathcal{M}, A \subseteq B\}$.

end while

if $1 \in \text{span}_{\mathbb{F}} \mathcal{S}$ **then**

OUTPUT **REFUTE**

else output **NOREFUTE**.

end if

³ Note that it is possible to derive a similar result independent of the parameter d . Since $d = 2$ for axioms of the form $\text{Ax}(\Gamma)$, the statement of Lemma 13 suffices for our purpose.

Note that to verify the condition of the **while** loop, one need not store in memory the set $\text{span}_{\mathbb{F}}\mathcal{S}$ (whose size is exponential in the input); this can be done by checking the solvability of linear equations. The number of iterations of the **while** loop is at most the number of multilinear monomials of degree at most k , thus ensuring that the procedure runs in polynomial time. Crucially, at each iteration of the **while** loop, the \mathbb{F} -span of \mathcal{S} has a *canonical* generating set.

Recall that we are assuming that $\mathbb{F} = \mathbb{Z}_p$. By viewing polynomials over \mathbb{Z}_p as vectors in the standard basis given by monomials, we encode P as a structure \mathfrak{A} with universe V over the vocabulary $\text{POLY}_p = (\text{Var}, \text{U}, \text{C}_0, \text{C}_1, \dots, \text{C}_{p-1})$, where:

1. $V = P \cup \{x_i \mid i \in [r]\} \cup \{1\}$.
2. Var and U are unary relational symbol with $\mathfrak{A} \models \text{Var}(v)$ if, and only if, $v \in \{x_i \mid i \in [r]\}$, and $\mathfrak{A} \models \text{U}(v)$ if, and only if, $v = 1$.
3. C_q for $q \in \mathbb{Z}_p$ are $(d+1)$ -ary relations, where d is the maximal degree of the polynomials in P . These encode P in matrix form; that is, $\mathfrak{A} \models \text{C}_q(u, v_1, \dots, v_d)$ if, and only if, $u \in P$, each v_i is equal to a variable or 1, and the coefficient of the monomial $v_1 v_2 \dots v_d$ in u is equal to q .

We now use a k -ary interpretation to obtain a structure whose universe is the set of multilinear polynomials of degree at most k . Formally, let $\text{POLY}_p^* = (\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{p-1}, \text{U}^*, \text{Mon}, \text{Sub})$ be a vocabulary where Mon and U^* are unary relational symbols, Sub is a binary relational symbol, and \mathbf{A}_q are as in the vocabulary LIN_p . One can then define an interpretation \mathcal{J} from POLY_p into POLY_p^* such that:

1. The universe of $\mathcal{J}(\mathfrak{A})$ are elements $\vec{v} \in V^k$ where either all v_i are equal to 1, all v_i are variables, or v_1 is a polynomial in P and v_2, \dots, v_k are either all equal to 1 or are variables such that $|\{v_i \mid 2 \leq i \leq k\}| \leq k - \deg(v_1)$. The relation $\equiv_{\mathcal{J}}$ partitions the universe into equivalence classes uniquely determined by the set of entries of each tuple. If the entries of \vec{v} are all equal to 1, we indicate its class by X_{\emptyset} , and similarly, if $v_i = x_{a_i}$ for all $i \in [k]$, we indicate its class by X_A where $A = \{a_i \mid i \in [k]\}$. If $v_1 = f$ for some $f \in P$ and v_2, \dots, v_k are all equal to 1, we indicate the class of \vec{v} by f and if $v_i = a_i$ for $2 \leq i \leq k$, we indicate its class by the pair (X_A, f) where $A = \{a_i \mid 2 \leq i \leq k\}$. We leave it to the reader to check that such an equivalence relation can be defined with first-order formulae.
2. $\mathfrak{A} \models \text{U}(\vec{v})$ and $\mathfrak{A} \models \text{Mon}(\vec{v})$ if, and only if, the equivalence class of \vec{v} is X_{\emptyset} and X_A with $|A| \geq 1$ respectively.
3. $\mathfrak{A} \models \mathbf{A}_q(\vec{u}, \vec{v})$ if, and only if, the equivalence class of \vec{u} and \vec{v} are X_A and (X_B, f) for some $f \in P$ respectively, and the coefficient of X_A in $X_B f$ equals q .
4. $\mathfrak{A} \models \text{Sub}(\vec{u}, \vec{v})$ if, and only if, the equivalence classes of \vec{u} and \vec{v} are X_A and X_B respectively and $A \subseteq B$.

The last thing required for proving Lemma 13 is showing the definability of monomials in the set \mathcal{S} after each iteration of the **while** loop. In what follows, set $\mathcal{T} = \{X_A f \mid f \in P, A \subseteq [r], \deg(X_A f) \leq k\}$.

Proof of Lemma 13. Let $\psi(z)$ be some FPS(p) formula over the vocabulary POLY_p^* . There is an interpretation $\mathcal{K}(t)$ of POLY_p^* into LIN_p such that $\mathcal{K}(\mathcal{J}(\mathfrak{A}), t \mapsto X_A)$ encodes the system of linear equations determining whether X_A is in the \mathbb{Z}_p -span of the set

$$\mathcal{T}_{\psi} = \mathcal{T} \cup \{X_B \mid (\mathfrak{A}, z \mapsto X_B) \models \psi(z) \wedge \text{Mon}(z)\}.$$

By Lemma 14, there is a FPS(p) formula $\theta_{\psi}(z)$, depending on ψ , such that $(\mathcal{J}(\mathfrak{A}), z \mapsto X_A) \models \theta_{\psi}(z)$ if, and only if, the monomial X_A is in the \mathbb{Z}_p -span of \mathcal{T}_{ψ} .

37:12 On the Relative Power of Linear Algebraic Approximations of Graph Isomorphism

In particular, replacing ψ with some unary relational variable Z ,

$$(\mathcal{J}(\mathfrak{A}), z \mapsto X_A) \models \exists y. \theta_Z(y) \wedge \text{Sub}(y, z)$$

holds if, and only if, there is some $B \subseteq A$ such that X_B is in the \mathbb{Z}_p -span of \mathcal{T}_Z . Whence,

$$(\mathcal{J}(\mathfrak{A}), z' \mapsto X_\emptyset) \models \text{if}_{\mathbb{Z}_p, z}(\exists y. \theta_Z(y) \wedge \text{Sub}(y, z))(z') \quad (2)$$

if, and only if, there is an MC_k refutation of P over \mathbb{Z}_p . By applying the Interpretation Lemma to the above formula and the interpretation \mathcal{J} , the desired result follows. \blacktriangleleft

Note that in formula 2, the solvability quantifier is included in the formula $\theta_Z(y)$.

► **Corollary 15.** *For any $k \in \mathbb{N}$, there is some k' such that for any graph Γ on V and $\vec{u}, \vec{v} \in V^k$, if there is a MC_k refutation of $\text{Ax}(\Gamma_{\vec{u} \rightarrow \vec{v}})$ over \mathbb{Z}_p , then there is some $\text{FOS}_{k'}(p)$ formula $\phi(\vec{z})$ distinguishing $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$.*

Proof. By Lemma 13 the equivalence classes of $\equiv_{\text{MC}_k}^p$ are definable in $\text{FPS}(p)$ and hence, by the embedding of $\text{FPS}(p)$ in infinitary $\text{FOS}(p)$, the statement follows. \blacktriangleleft

Combining the latter with Theorem 10 and Lemma 7, one deduces Theorem 1.

5 Nullstellensatz refutations and S_k^p -stability

The focus of this section is the proof of the following statement.

► **Lemma 16.** *Let Γ be a graph on V and let $\gamma \in \mathcal{P}(V^k)$. If for all $\vec{u}, \vec{v} \in V^k$, $\gamma(\vec{u}) = \gamma(\vec{v})$ if, and only if, $\vec{u} \equiv_{\text{NC}_k}^p \vec{v}$, then γ is S_k^p -stable.*

Theorems 2 and 3 are its direct consequences.

5.1 Proof of Lemma 16

In what follows, $\gamma \in \mathcal{P}(V^k)$ satisfies the assumptions of Lemma 16, $\vec{u}, \vec{v} \in V^k$ and $\vec{\chi}$ and $\vec{\xi}$ are the (\vec{i}, \vec{u}) and (\vec{i}, \vec{v}) -character vectors of γ respectively, where we may assume without loss of generality that $\vec{i} = (k, k-1, \dots, k-2r+1) \in [k]^{(2r)}$, for some r with $2r \leq k$. For $\vec{w}, \vec{z} \in V^l$, denote by $X_{\vec{w}\vec{z}}$ the monomial $x_{w_1 z_1} x_{w_2 z_2} \dots x_{w_l z_l}$ (which need not be multilinear).

Recall that γ is $S_{k,r}^p$ -stable, if, and only if, for every \vec{u}, \vec{v} there is some matrix $T \in \mathfrak{P}_{V^r}(\mathbb{Z}_p)$ such that $\chi_\sigma T = T \xi_\sigma$, for all $\sigma \in \text{Im}(\gamma)$. That is, the following system of equations is solvable in the variables $T_{\vec{w}\vec{z}}$:

$$\sum_{\vec{a} \in V^r} (\chi_\sigma)_{\vec{w}\vec{a}} T_{\vec{a}\vec{z}} - \sum_{\vec{a} \in V^r} T_{\vec{w}\vec{a}} (\xi_\sigma)_{\vec{a}\vec{z}} = 0 \quad \text{for } \sigma \in \text{Im}(\gamma), \vec{w}, \vec{z} \in V^r$$

$$\sum_{\vec{a} \in V^r} T_{\vec{w}\vec{a}} - 1 = 0 \quad \text{and} \quad \sum_{\vec{a} \in V^r} T_{\vec{a}\vec{z}} - 1 = 0 \quad \text{for } \vec{w}, \vec{z} \in V^r,$$

where $(\chi_\sigma)_{\vec{w}\vec{a}}$ is equal to 1 if $\gamma(\vec{u} \langle \vec{i}, \vec{w} \cdot \vec{a} \rangle) = \sigma$ and 0 otherwise (similarly for ξ_σ).

We show that there is a NC_k derivation from $\text{Ax}(\Gamma_{\vec{v} \rightarrow \vec{u}})$ over \mathbb{Z}_p of the following multilinear polynomials (Lemma 19):

$$\sum_{\vec{a} | \gamma(\vec{u}(\vec{i}, \vec{w} \cdot \vec{a})) = \sigma} X_{\vec{a}\vec{z}} - \sum_{\vec{a} | \gamma(\vec{v}(\vec{i}, \vec{a} \cdot \vec{z})) = \sigma} X_{\vec{w}\vec{a}} \text{ for } \sigma \in \text{Im}(\gamma), \vec{w}, \vec{z} \in V^r \quad (3)$$

$$\sum_{\vec{a} \in V^r} X_{\vec{w}\vec{a}} - 1 \text{ and } \sum_{\vec{a} \in V^r} X_{\vec{a}\vec{z}} - 1 \text{ for } \vec{w}, \vec{z} \in V^r. \quad (4)$$

We may view each monomial (apart from the constant term) as a distinct linear variable, so that all of the above are linear polynomials. Since $\gamma(\vec{u}) = \gamma(\vec{v})$, there is no NC_k refutation of $\text{Ax}(\Gamma_{\vec{v} \rightarrow \vec{u}})$ and hence, no linear combination of 3 and 4 gives the constant polynomial 1. It follows that if viewed as linear polynomials, 3 and 4 have a common root, thus showing Lemma 16.

► **Lemma 17.** *If $\gamma(\vec{u}(\vec{i}, \vec{w} \cdot \vec{z})) \neq \gamma(\vec{v}(\vec{i}, \vec{w}' \cdot \vec{z}'))$, then there is a NC_k derivation over \mathbb{Z}_p from $\text{Ax}(\Gamma_{\vec{v} \rightarrow \vec{u}})$ of $X_{\vec{w}\vec{w}'} X_{\vec{z}\vec{z}'}$.*

Proof. Set $Y = X_{\vec{w}\vec{w}'} X_{\vec{z}\vec{z}'}$ and let $X_{\vec{u}'\vec{v}'}$ be the degree $k - 2r$ monomial where $u'_j = u_j$ and $v'_j = v_j$ for $j \in [k - 2r]$. Then, there is a NC_k derivation of $Y(X_{\vec{u}'\vec{v}'} - 1)$, for indeed

$$Y(X_{\vec{u}'\vec{v}'} - 1) = Y(x_{u_1 v_1} - 1) + Y x_{u_1 v_1} (x_{u_2 v_2} - 1) + \dots + Y (x_{u_1 v_1} \dots x_{u_{k-2r-1} v_{k-2r-1}}) (x_{u_{k-2r} v_{k-2r}} - 1).$$

By assumption, $\gamma(\vec{u}(\vec{i}, \vec{w} \cdot \vec{z})) \neq \gamma(\vec{v}(\vec{i}, \vec{w}' \cdot \vec{z}'))$ and hence, there is an NC_k derivation of $Y X_{\vec{u}'\vec{v}'}$. Subtracting the latter from $Y(X_{\vec{u}'\vec{v}'} - 1)$ yields the desired statement. ◀

► **Lemma 18.** *For any $\vec{w}, \vec{z} \in V^r$ and $\vec{s} \in V^t$ there is a NC_{t+r} derivation over \mathbb{Z}_p from $\text{Ax}(\Gamma_{\vec{v} \rightarrow \vec{u}})$ of*

$$X_{\vec{w}\vec{z}} \left(\sum_{\vec{a} \in V^t} X_{\vec{s}\vec{a}} - 1 \right) \text{ and } X_{\vec{w}\vec{z}} \left(\sum_{\vec{a} \in V^t} X_{\vec{a}\vec{s}} - 1 \right).$$

Proof. We proceed by induction on t . For $t = 1$, $X_{\vec{w}\vec{z}}(\sum_{a \in V} x_{sa} - 1)$ is the product of a monomial and an axiom, so has a NC_{r+1} derivation.

Assume $X_{\vec{w}\vec{z}}(\sum_{\vec{a} \in V^t} X_{\vec{s}\vec{a}} - 1)$ has a NC_{t+r} derivation. It can be easily verified that if a polynomial f has a NC_r derivation from some set of axioms, then xf has a NC_{r+1} derivation for any variable x . Thus, $x_{s'a'} X_{\vec{w}\vec{z}}(\sum_{\vec{a} \in V^t} X_{\vec{s}\vec{a}} - 1)$ has a NC_{t+r+1} derivation. Finally

$$\sum_{a' \in V} x_{s'a'} X_{\vec{w}\vec{z}} \left(\sum_{\vec{a} \in V^t} X_{\vec{s}\vec{a}} - 1 \right) = X_{\vec{w}\vec{z}} \left(\sum_{\vec{a} \in V^t, a' \in V} X_{(\vec{s}, s')(\vec{a}, a')} - 1 \right) = X_{\vec{w}\vec{z}} \left(\sum_{\vec{a} \in V^{t+1}} X_{(\vec{s}, s')\vec{a}} - 1 \right)$$

as required. ◀

► **Lemma 19.** *There is a NC_k derivation over \mathbb{Z}_p from $\text{Ax}(\Gamma_{\vec{v} \rightarrow \vec{u}})$ of the polynomials in formulae (3) and (4).*

Proof. For $\vec{a} \in V^r$, set $\mathcal{N}(\vec{u}, \vec{w}) = \{\vec{a} \in V^r \mid \gamma(\vec{u}(\vec{i}, \vec{w} \cdot \vec{a})) = \sigma\}$ and $\mathcal{N}(\vec{v}, \vec{z}) = \{\vec{a} \in V^r \mid \gamma(\vec{v}(\vec{i}, \vec{a} \cdot \vec{z})) = \sigma\}$ (note the slight asymmetry). By Lemma 18, there is a NC_{2r} (and hence, NC_k , since $2r \leq k$) derivation of $X_{\vec{a}\vec{z}}(\sum_{\vec{a}' \in V^r} X_{\vec{s}\vec{a}'} - 1)$ for every $\vec{a}, \vec{s}, \vec{z} \in V^r$. By subtracting from the above all monomials $X_{\vec{a}\vec{z}} X_{\vec{s}\vec{a}'}$ for which $\vec{a}' \notin \mathcal{N}(\vec{v}, \vec{z})$ (which have a NC_k derivation by Lemma 17) one gets $X_{\vec{a}\vec{z}}(\sum_{\vec{a}' \in \mathcal{N}(\vec{v}, \vec{z})} X_{\vec{s}\vec{a}'} - 1)$. Adding these for all $\vec{a} \in \mathcal{N}(\vec{u}, \vec{w})$ yields

$$\sum_{\vec{a} \in \mathcal{N}(\vec{u}, \vec{w})} X_{\vec{a}\vec{z}} \left(\sum_{\vec{a}' \in \mathcal{N}(\vec{v}, \vec{z})} X_{\vec{s}\vec{a}'} - 1 \right). \quad (5)$$

A similar argument shows that there is a NC_k derivation of

$$\sum_{\vec{a} \in \mathcal{N}(\vec{v}, \vec{z})} X_{\vec{w}\vec{a}} \left(\sum_{\vec{a}' \in \mathcal{N}(\vec{u}, \vec{w})} X_{\vec{a}'\vec{s}} - 1 \right). \quad (6)$$

Subtracting (5) from (6) yields (3).

The polynomials in (4) can be derived by setting $r = 0$ in Lemma 18. \blacktriangleleft

5.2 Generalized Cai-Fürer-Immerman constructions

In 1992, Cai, Fürer and Immerman provided, for $k \geq 1$, a family of pairs of non-isomorphic graphs $(\mathcal{G}_k, \mathcal{H}_k)$ which cannot be distinguished by \mathcal{C}_k -formulae. These graphs really encode a system of linear equations over \mathbb{Z}_2 , the solvability of which can be decided in polynomial time. These structures can be generalized to encode a systems of linear equations over an arbitrary finite field. Loosely speaking, the *generalized Cai-Fürer-Immerman* construction for the field \mathbb{Z}_p provides, for each $k \in \mathbb{N}$, p non-isomorphic 3-regular graphs $\mathcal{G}_k^{(1)}, \dots, \mathcal{G}_k^{(p)}$. These delimit the power of well known linear algebra based polynomial-time approximations of graph isomorphism. Let $\Gamma_{k,p}$ be the disjoint union of the graphs $\mathcal{G}_k^{(1)}, \dots, \mathcal{G}_k^{(p)}$ and let V denote its vertex set.

► **Theorem 20** (Theorems 8.1 and 8.2 in [10]). *If $\Gamma = \Gamma_{k,p}$ and $q \neq p$, then the equivalence classes of $[\alpha_{k,\Gamma}]^{\text{IM}_k^q}$ do not coincide with those of the k -orbit partition for Γ . If $q = p$, the equivalence classes of $[\alpha_{3,\Gamma}]^{\text{IM}_3^q}$ coincide with those of the 3-orbit partition for Γ .*

Originally, the generalized Cai-Fürer-Immerman constructions were introduced to delimit the expressive power of the extension of fixed point logics with rank operators over a finite field $\text{FPR}(p)$ and, correspondingly, the distinguishing power of the extension of first order logic by said operators, $\text{FOR}(p)$ (see Chapter 7 of [18]). The distinguishing power of the invertible map test in characteristic p is at least that of $\text{FOR}(p)$, so in one direction Theorem 20 provides a lower bound for this logic. In the other direction, we can show that the orbit partition on $\Gamma_{k,p}$ can already be defined in $\text{FPR}(p)$. Indeed, it can be defined in the apparently weaker logic $\text{FPS}(p)$, giving the following result, of which Theorem 3 is a direct consequence.

► **Theorem 21.** *Let $\Gamma = \Gamma_{k,p}$. If $p \neq q$, there exist $\vec{u}, \vec{v} \in V^k$ in different equivalence classes of the k -orbit partition of Γ , such that there are no $\text{FOS}_k(q)$ -formulae distinguishing $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$. If $q = p$ and $\vec{u}, \vec{v} \in V^2$, then $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{u})$ is distinguished from $(\mathfrak{A}_\Gamma, \vec{z} \mapsto \vec{v})$ by some $\text{FOS}_2(q)$ -formula if, and only if, \vec{u}, \vec{v} are in different equivalence classes of the 2-orbit partition of Γ .*

6 Conclusions: where does polynomial calculus lie?

The invertible map tests can be thought of a family of algorithms, each of which distinguishes tuples of vertices of graphs according to the most general linear algebraic invariants expressible with a bounded number of variables in a logic. As bounded degree PC, MC and NC refutations can be decided solely by using basic field operations, one expects that the equivalences defined by the invertible map tests simulate those defined by the above mentioned proof systems. Theorem 1 gives a partial proof of this conjecture, leaving it open as to whether the invertible map tests can simulate bounded degree PC refutations, when taken over some finite field.

Our approach to this question was to attempt to define the above proof systems in the simplest logics which could express the solvability of systems of linear equations. The proof of Lemma 13 hints at the flaws of this choice. Deciding whether or not there is a PC_k refutation of $P \subseteq \mathbb{F}[x_1, \dots, x_r]$ can be understood as the following procedure, similar to that in Section 4.2.

INPUT: $P \subset \mathbb{F}[x_1, \dots, x_r]$
 OUTPUT: REFUTE or NOREFUTE.
 Initialize $\mathcal{S} = \{X_A f \mid f \in P, \deg(X_A f) \leq k\}$.
while $\text{span}_{\mathbb{F}}\mathcal{S}$ has changed since last round or $1 \notin \text{span}_{\mathbb{F}}\mathcal{S}$ **do**
 Find a set \mathcal{B} generating the \mathbb{F} -space $\{f \in \text{span}_{\mathbb{F}}\mathcal{S} \mid \deg(f) < k\}$.
 $\mathcal{S} \leftarrow \mathcal{S} \cup \{X_A f \mid f \in \mathcal{B}, \deg(X_A f) \leq k\}$.
end while
if $1 \in \text{span}_{\mathbb{F}}\mathcal{S}$ **output** REFUTE. **then**
else **output** NOREFUTE
end if

This procedure runs in polynomial time, as one can find \mathcal{B} by using Gaussian elimination (there is no need to store the set $\text{span}_{\mathbb{F}}\mathcal{S}$ as a generating set suffices), and the number of iterations of the **while** loop is bounded by the number of monomials of degree at most k in the variables $\{x_1, \dots, x_r\}$. If \mathbb{F} is finite, this procedure is a priori *not* definable in FPS(p), as it is not immediate whether there is a canonical choice for the set \mathcal{B} (its counterpart in the procedure for monomial calculus refutations was the set \mathcal{M} of monomials in the span of \mathcal{S}). Put otherwise, defining the set \mathcal{B} requires defining the solution space of a system of linear equations over a field of positive characteristic p , rather than just determining the solvability of the system and it is not clear if this can be done in FPS(p). The FPC definability of bounded degree polynomial calculus over the field \mathbb{Q} (Theorem 4.9 in [17]) relies in fact on the FPC definability of solution spaces of linear equations over \mathbb{Q} (Theorem 4.11 in [17]).

Let us view the problem from the viewpoint of proof complexity. It follows from Lemma 16, that if PC_k refutations over \mathbb{Z}_p are definable in FPS(p), then for every k , there is some k' such that if $\text{Ax}(\Gamma_{\vec{u} \rightarrow \vec{v}})$ has a PC_k refutation over \mathbb{Z}_p , then it has a $\text{NC}_{k'}$ refutation over \mathbb{Z}_p . It is known that for all n and for any field, there is a set of axioms on $n(n+1)$ variables which can be refuted by PC_3 but require degree $\Omega(n)$ to be refuted by NC (Theorem 6 in [5]). Furthermore, this lower bound can be shown to be optimal. On the other hand, Buss et al. have shown that NC derivations can be used to simulate *tree-like* PC derivations (see Theorems 5.3 and 5.4 in [6]) with only a small increase in degree. For a set of axioms of the form $\text{Ax}(\Gamma_{\vec{u} \rightarrow \vec{v}})$, it is not known if any PC_k refutation of such can be converted into a tree-like refutation without incurring in an unbounded increase in degree.

References

- 1 A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666–1683, 2009. Automata, Languages and Programming (ICALP 2007).
- 2 L. Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proc. 48th Annual ACM SIGACT Symp. Theory of Computing, STOC*, pages 684–697, 2016.
- 3 P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and P. Pudlak. Lower bounds on Hilbert’s Nullstellensatz and propositional proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, volume 73, pages 794–806, 1994.
- 4 C. Berkholz and M. Grohe. Limitations of algebraic approaches to graph isomorphism testing. *CoRR*, abs/1502.05912, 2015.
- 5 S. R. Buss. Lower bounds on Nullstellensatz proofs via designs. In *in Proof Complexity and Feasible Arithmetics, P. Beame and S. Buss, eds., American Mathematical Society*, pages 59–71. American Math. Soc, 1998.
- 6 S. R. Buss, R. Impagliazzo, J. Krajíček, P. Pudlak, A. Razborov, and J. Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computational Complexity*, 6:256–298, January 1997. doi:10.1007/BF01294258.

- 7 J. Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.
- 8 M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Gröbner basis algorithm to find proofs of unsatisfiability. *Proceedings of STOC'96*, March 2000.
- 9 A. Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, 2:8–21, 2015.
- 10 A. Dawar, E. Grädel, and W. Pakusa. Approximations of isomorphism and logics with linear algebraic operators. In *46th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 112:1–112:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.112.
- 11 A. Dawar, E. Graedel, B. Holm, E. Kopczyński, and W. Pakusa. Definability of linear equation systems over groups and rings. *Logical Methods in Computer Science*, 9, April 2012. doi:10.2168/LMCS-9(4:12)2013.
- 12 A. Dawar and B. Holm. Pebble games with algebraic rules. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, pages 251–262, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- 13 A. Dawar and D. Vagnozzi. Generalizations of k -Weisfeiler-Leman stabilization. *Moscow Journal of Number Theory and Combinatorics*, 2020.
- 14 A. Dawar and D. Vagnozzi. On the relative power of algebraic approximations of graph isomorphism. *arXiv*, 2021. arXiv:2103.16294.
- 15 C. de Seguins Pazzis. Invariance of simultaneous similarity and equivalence of matrices under extension of the ground field. *Linear Algebra and its Applications*, 433, February 2009. doi:10.1016/j.laa.2010.03.022.
- 16 H-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 1999.
- 17 E. Grädel, M. Grohe, B. Pago, and W. Pakusa. A finite-model-theoretic view on propositional proof complexity. *Logical Methods in Computer Science*, 15(1), 2019.
- 18 B. Holm. *Descriptive Complexity of Linear Algebra*. PhD thesis, University of Cambridge, 2010.
- 19 M. Lichter. Separating rank logic from polynomial time. In *Proc. 36th ACM/IEEE Symp. on Logic in Computer Science (LICS)*, 2021.
- 20 R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8:131–136, 1979.
- 21 M. Otto. *Bounded Variable Logics and Counting – A Study in Finite Models*, volume 9 of *Lecture Notes in Logic*. Springer-Verlag, 1997.
- 22 G. Tinhofer. Graph isomorphism and theorems of Birkhoff type. *Computing*, 36:285–300, 1986.


Maximum Cut on Interval Graphs of Interval Count Four Is NP-Complete

Celina M. H. de Figueiredo ✉ 

Federal University of Rio de Janeiro, Brazil

Alexsander A. de Melo ✉ 

Federal University of Rio de Janeiro, Brazil

Fabiano S. Oliveira ✉ 

Rio de Janeiro State University, Brazil

Ana Silva ✉ 

Federal University of Ceará, Brazil

Abstract

The computational complexity of the MAXCUT problem restricted to interval graphs has been open since the 80's, being one of the problems proposed by Johnson on his *Ongoing Guide to NP-completeness*, and has been settled as NP-complete only recently by Adhikary, Bose, Mukherjee and Roy. On the other hand, many flawed proofs of polynomiality for MAXCUT on the more restrictive class of unit/proper interval graphs (or graphs with interval count 1) have been presented along the years, and the classification of the problem is still not known. In this paper, we present the first NP-completeness proof for MAXCUT when restricted to interval graphs with bounded interval count, namely graphs with interval count 4.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Mathematics of computing → Graph theory

Keywords and phrases maximum cut, interval graphs, interval lengths, interval count, NP-complete

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.38

Related Version *Full Version:* <https://arxiv.org/abs/2012.09804>

Funding *Celina M. H. de Figueiredo:* Partially funded by CNPq grants 302823/2016-6 and 407635/2018-1, CAPES Finance Code 001, and FAPERJ CNE E-26/202.793/2017.

Alexsander A. de Melo: Partially funded by CNPq grant 140399/2017-8 and CAPES Finance Code 001.

Fabiano S. Oliveira: Partially funded by FAPERJ grant E-26/010.002239/2019.

Ana Silva: Partially funded by CNPq grants 303803/2020-7 and 437841/2018-9, and FUNCAP/CNPq grant 0112-00061.01.00/16.

Acknowledgements We thank Vinicius F. Santos who shared Reference [1], and anonymous referees for many valuable suggestions, including improving the interval count from 5 to 4.

1 Introduction

A *cut* is a partition of the vertex set of a graph into two disjoint parts and the *maximum cut problem* (denoted MAXCUT for short) aims to determine a cut with the maximum number of edges for which each endpoint is in a distinct part. The decision problem MAXCUT is known to be NP-complete since the seventies [15], and only recently its restriction to interval graphs has been announced to be hard [1], settling a long-standing open problem that appeared in the 1985 column of the *Ongoing Guide to NP-completeness* by David S. Johnson [17]. We refer the reader to a revised version of the table in [12], where one can also find a parameterized complexity version of said table.



© Celina M. H. de Figueiredo, Alexsander A. de Melo, Fabiano S. Oliveira, and Ana Silva; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 38; pp. 38:1–38:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

An *interval model* is a family of closed intervals of the real line. A graph is an *interval graph* if there exists an interval model, for which each interval corresponds to a vertex of the graph, such that distinct vertices are adjacent in the graph if and only if the corresponding intervals intersect. Ronald L. Graham proposed in the 80's the study of the *interval count* of an interval graph as the smallest number of interval lengths used by an interval model of the graph. Interval graphs having interval count 1 are called *unit intervals* (can also be called proper interval, or indifference). Understanding the interval count, besides being an interesting and challenging problem by itself, can be also of value for the investigation of problems that are hard for general interval graphs, and easy for unit interval graphs (e.g. geodetic number [8,13], optimal linear arrangement [9,16], sum coloring [20,21]). The positive results for unit interval graphs usually take advantage of the fact that a representation for these graphs can be found in linear time [10,11]. Surprisingly, the recognition of interval graphs with interval count k is open, even for $k = 2$ [7]. Nevertheless, another generalization of unit interval graphs has been recently introduced which might be more promising in this aspect. These graphs are called *k-nested interval graphs*, introduced in [18], where the authors, among other things, give a linear time recognition algorithm.

In the same way that MAXCUT on interval graphs has evaded being solved for so long, the community has been puzzled by the restriction to unit interval graphs. Indeed, two attempts at solving it in polynomial time were proposed in [4,6] just to be disproved closely after [3,19]. In this paper, we give the first classification that bounds the interval count, namely, we prove that MAXCUT is NP-complete when restricted to interval graphs of interval count 4. This also implies NP-completeness for the newly generalized class of 4-nested graphs, and opens the search for a full polynomial/NP-complete dichotomy classification in terms of the interval count. It can still happen that the problem is hard even on graphs of interval count 1. We contribute towards filling the complexity gap between interval and unit interval graphs.

Next, we establish basic definitions and notation. Section 2 describes our reduction and Section 3 discusses the interval count of the interval graph constructed in [1].

1.1 Preliminaries

In this work, all graphs considered are simple. For missing definitions and notation of graph theory, we refer to [5]. For a comprehensive study of interval graphs, we refer to [14].

Let G be a graph. Let X and Y be two disjoint subsets of $V(G)$. We let $E_G(X, Y)$ be the set of edges of G with an endpoint in X and the other endpoint in Y . For every subset $S \subseteq V(G)$, we let $S^X = S \cap X$ and $S^Y = S \cap Y$. A *cut* of G is a partition of $V(G)$ into two parts $A, B \subseteq V(G)$, denoted by $[A, B]$; the edge set $E_G(A, B)$ is called the *cut-set* of G associated with $[A, B]$. For each two vertices $u, v \in V(G)$, we say that u and v are in a *same part* of $[A, B]$ if either $\{u, v\} \subseteq A$ or $\{u, v\} \subseteq B$; otherwise, we say that u and v are in *opposite parts* of $[A, B]$. Denote by $\text{mc}(G)$ the maximum size of a cut-set of G . The MAXCUT problem has as input a graph G and a positive integer k , and it asks whether $\text{mc}(G) \geq k$.

Let $I \subseteq \mathbb{R}$ be a closed interval of the real line. We let $\ell(I)$ and $r(I)$ denote respectively the minimum and maximum points of I , which we call the *left* and the *right endpoints* of I , respectively. We denote a closed interval I by $[\ell(I), r(I)]$. The *length* of an interval I is defined as $|I| = r(I) - \ell(I)$. An *interval model* is a finite multiset \mathcal{M} of intervals. The *interval count* of an interval model \mathcal{M} , denoted by $\text{ic}(\mathcal{M})$, is defined as the number of distinct lengths of the intervals in \mathcal{M} . Let G be a graph and \mathcal{M} be an interval model. An \mathcal{M} -*representation* of G is a bijection $\phi: V(G) \rightarrow \mathcal{M}$ such that, for every two distinct vertices $u, v \in V(G)$, we have that $uv \in E(G)$ if and only if $\phi(u) \cap \phi(v) \neq \emptyset$. If such an \mathcal{M} -representation exists, we

say that \mathcal{M} is an *interval model* of G . We note that a graph may have either no interval model or arbitrarily many distinct interval models. A graph is called an *interval graph* if it has an interval model. The *interval count* of an interval graph G , denoted by $\text{ic}(G)$, is defined as $\text{ic}(G) = \min\{\text{ic}(\mathcal{M}) : \mathcal{M} \text{ is an interval model of } G\}$. An interval graph is called a *unit interval graph* if its interval count is equal to 1.

Note that, for every interval model \mathcal{M} , there exists a unique (up to isomorphism) graph that admits an \mathcal{M} -representation. Thus, for every interval model $\mathcal{M} = \{I_1, \dots, I_n\}$, we let $\mathbb{G}_{\mathcal{M}}$ be the graph with vertex set $V(\mathbb{G}_{\mathcal{M}}) = \{1, \dots, n\}$ and edge set $E(\mathbb{G}_{\mathcal{M}}) = \{ij : I_i, I_j \in \mathcal{M}, I_i \cap I_j \neq \emptyset, i \neq j\}$. Since $\mathbb{G}_{\mathcal{M}}$ is uniquely determined (up to isomorphism) from \mathcal{M} , in what follows we may make an abuse of language and use graph terminologies to describe properties related to the intervals in \mathcal{M} . Two intervals $I_i, I_j \in \mathcal{M}$ are said to be *true twins* in $\mathbb{G}_{\mathcal{M}}$ if they have the same close neighborhood in $\mathbb{G}_{\mathcal{M}}$, i.e. $N_{\mathbb{G}_{\mathcal{M}}}(I_i) \cup \{I_i\} = N_{\mathbb{G}_{\mathcal{M}}}(I_j) \cup \{I_j\}$.

For each three positive integers $a, b, c \in \mathbb{N}_+$, we write $a \equiv_b c$ to denote that a modulo b is equal to c modulo b .

2 Our reduction

The following theorem is the main contribution of this work:

► **Theorem 1.** *MAXCUT is NP-complete on interval graphs of interval count 4.*

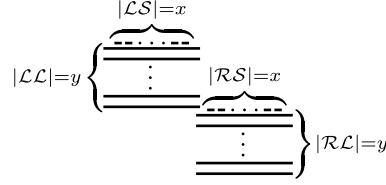
This result is a stronger version of that of Adhikary et al. [1]. To prove Theorem 1, we present a polynomial-time reduction from MAXCUT on cubic graphs, which is known to be NP-complete [2]. In order to explain the technical effort needed to push the construction of Adhikary et al. enabling our construction of a reduction graph that uses only four different lengths of intervals, we present our construction in three sections. First, we explain how the key gadget of Adhikary et al. relates the number of intervals of each size to the part where they are placed in a maximum cut. Second, we present our new gadget that organizes copies of the original key gadget into an escalator grid, which constitutes our key gadget to obtain a reduction graph that admits a model with an interval count bounded by a constant. Third, an outline of the proof explains how our use of the base gadgetry due to Adhikary et al. through the escalator allows us to relate maximum cuts of the input graph to maximum cuts of the reduction graph.

2.1 Grained gadget

The interval graph constructed in the reduction of [1] is strongly based on two types of gadgets, called *V-gadgets* and *E-gadgets*. In fact, these gadgets are the same, except for the amount of intervals of certain kinds contained in each of them. In this subsection, we present a generalization of such gadgets, rewriting their key properties to suit our purposes. In order to discuss the interval count of the reduction of [1], we describe it in details in Section 3.

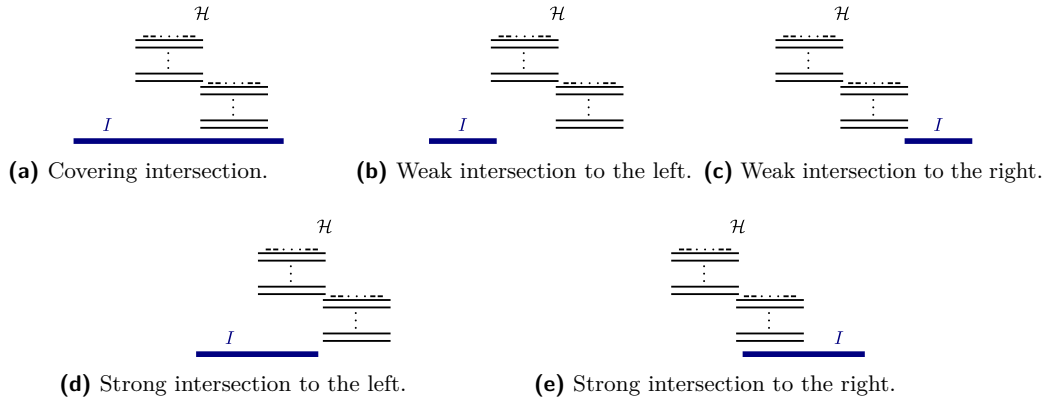
Let x and y be two positive integers. An (x, y) -*grained gadget* is an interval model \mathcal{H} formed by y long intervals (called *left long*) intersecting in their right endpoint with other y long intervals (called *right long*), together with $2x$ short intervals, x of which intersect exactly the y left long ones (called *left short*), and x of which intersect exactly the y right long ones (called *right short*); see Figure 1. We write $\mathcal{LS}(\mathcal{H})$, $\mathcal{LL}(\mathcal{H})$, $\mathcal{RS}(\mathcal{H})$ and $\mathcal{RL}(\mathcal{H})$ to denote the left short, left long, right short and right long intervals of \mathcal{H} , respectively, and we omit \mathcal{H} when it is clear from the context.

Note that, if \mathcal{H} is an (x, y) -grained gadget, then $\mathbb{G}_{\mathcal{H}}$ is a split graph such that $\mathcal{LS} \cup \mathcal{RS}$ is an independent set of size $2x$, $\mathcal{LL} \cup \mathcal{RL}$ is a clique of size $2y$, $N_{\mathbb{G}_{\mathcal{H}}}(\mathcal{LS}) = \mathcal{LL}$ and $N_{\mathbb{G}_{\mathcal{H}}}(\mathcal{RS}) = \mathcal{RL}$. Moreover, the intervals in \mathcal{LL} are true twins in $\mathbb{G}_{\mathcal{H}}$; similarly, the intervals in \mathcal{RL} are true twins in $\mathbb{G}_{\mathcal{H}}$.



■ **Figure 1** General structure of an (x, y) -grained gadget.

Let \mathcal{M} be an interval model containing an (x, y) -grained gadget \mathcal{H} . The possible types of intersections between an interval $I \in \mathcal{M} \setminus \mathcal{H}$ and \mathcal{H} in our construction are depicted in Figure 2, using our notation. More specifically, the *cover intersection* intersects all the intervals, the *weak intersection to the left (right)* intersects exactly the left (right) long intervals, while the *strong intersection to the left (right)* intersects exactly the left (right) long and short intervals. We say that \mathcal{M} *respects the structure* of \mathcal{H} if I either does not intersect \mathcal{H} at all, or intersects \mathcal{H} as depicted in Figure 2.



■ **Figure 2** (a) Interval $I \in \mathcal{M} \setminus \mathcal{H}$ covering \mathcal{H} , (b-c) weakly intersecting \mathcal{H} to the left and to the right, and (d-e) strongly intersecting \mathcal{H} to the left and to the right.

The advantage of this gadget is that, by manipulating the values of x and y , we can ensure that, in a maximum cut, the left long and right short intervals are placed in the same part, opposite to the part containing the left short and right long intervals. The next lemma is a step in this direction. Denote by $c_{\mathcal{M}}(\mathcal{H})$ the number of intervals of \mathcal{M} that cover \mathcal{H} ; by $wkl_{\mathcal{M}}(\mathcal{H})$ (resp. $wkr_{\mathcal{M}}(\mathcal{H})$) the number of intervals of \mathcal{M} that weakly intersect \mathcal{H} to the left (resp. right); and by $stl_{\mathcal{M}}(\mathcal{H})$ (resp. $str_{\mathcal{M}}(\mathcal{H})$) the number of intervals of \mathcal{M} that strongly intersect \mathcal{H} to the left (resp. right).

► **Lemma 2.** *Let x and y be positive integers, \mathcal{H} be an (x, y) -grained gadget and \mathcal{M} be an interval model that respects the structure of \mathcal{H} . For every maximum cut $[A, B]$ of $\mathbb{G}_{\mathcal{M}}$, the following conditions hold:*

1. *if $y + stl_{\mathcal{M}}(\mathcal{H}) + c_{\mathcal{M}}(\mathcal{H}) \equiv_2 1$ and $x > 2y - 1 + wkl_{\mathcal{M}}(\mathcal{H}) + stl_{\mathcal{M}}(\mathcal{H}) + c_{\mathcal{M}}(\mathcal{H})$, then $\mathcal{LS}(\mathcal{H}) \subseteq A$ and $\mathcal{LL}(\mathcal{H}) \subseteq B$, or vice versa;*
2. *if $y + str_{\mathcal{M}}(\mathcal{H}) + c_{\mathcal{M}}(\mathcal{H}) \equiv_2 1$ and $x > 2y - 1 + wkr_{\mathcal{M}}(\mathcal{H}) + str_{\mathcal{M}}(\mathcal{H}) + c_{\mathcal{M}}(\mathcal{H})$, then $\mathcal{RS}(\mathcal{H}) \subseteq A$ and $\mathcal{RL}(\mathcal{H}) \subseteq B$, or vice versa.*

Now, we want to add conditions that, together with the ones from the previous lemma, ensure that the left long intervals will be put opposite to the right long intervals. Based on Lemma 2, we say that $(\mathcal{H}, \mathcal{M})$ is *well-valued* if Conditions (1) and (2) hold, in addition to the following inequality

$$y^2 > y \cdot \text{wkr}_{\mathcal{M}}(\mathcal{H}) + (x - y) \cdot (\text{str}_{\mathcal{M}}(\mathcal{H}) + c_{\mathcal{M}}(\mathcal{H})). \quad (1)$$

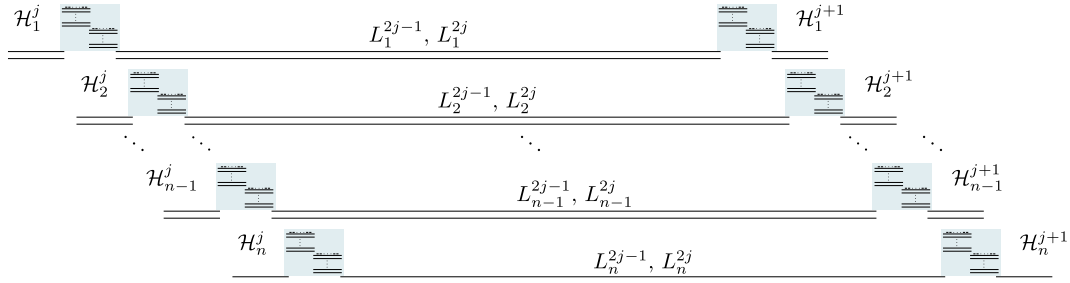
Let $[A, B]$ be a maximum cut of $\mathbb{G}_{\mathcal{M}}$. We say that \mathcal{H} is *A-partitioned* by $[A, B]$ if $\mathcal{L}\mathcal{S}(\mathcal{H}) \cup \mathcal{R}\mathcal{L}(\mathcal{H}) \subseteq A$, and $\mathcal{R}\mathcal{S}(\mathcal{H}) \cup \mathcal{L}\mathcal{L}(\mathcal{H}) \subseteq B$. Define *B-partitioned* analogously. The next lemma finally ensures what we wanted.

► **Lemma 3.** *Let x and y be positive integers, \mathcal{H} be an (x, y) -grained gadget, \mathcal{M} be an interval model and $[A, B]$ be a maximum cut of $\mathbb{G}_{\mathcal{M}}$. If \mathcal{M} respects the structure of \mathcal{H} and $(\mathcal{H}, \mathcal{M})$ is well-valued, then \mathcal{H} is either A-partitioned or B-partitioned by $[A, B]$.*

We have rewritten above in a more technical form the lemmas presented in [1], so that we are able to explicitly give the conditions that ensure the key property of their gadgets.

2.2 Reduction graph

In this subsection, we formally present our construction. Recall that we are making a reduction from MAXCUT on cubic graphs. So, consider a cubic graph G on n vertices and m edges. Intuitively, we consider an ordering of the edges of G , and we divide the real line into m regions, with the j -th region holding the information about whether the j -th edge is in the cut-set. For this, each vertex u will be related to a subset of intervals traversing all the m regions, bringing the information about which part u belongs to. Let $\pi_V = (v_1, \dots, v_n)$ be an ordering of $V(G)$, $\pi_E = (e_1, \dots, e_m)$ be an ordering of $E(G)$, and $\mathfrak{G} = (G, \pi_V, \pi_E)$.

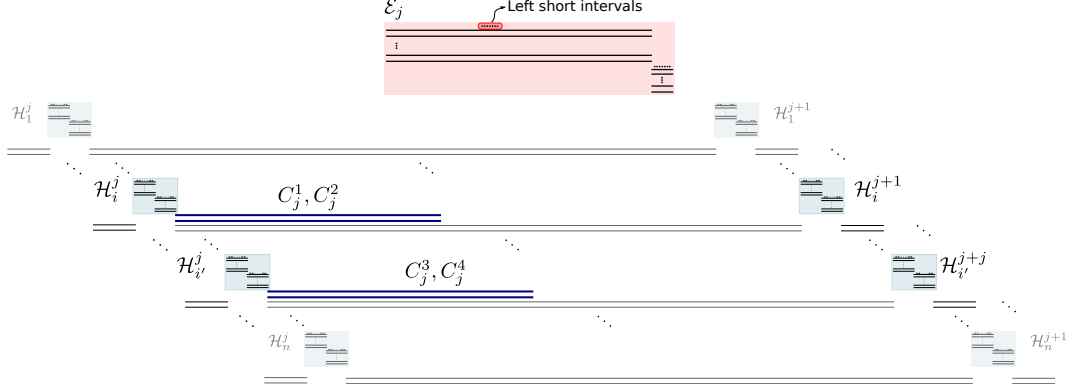


■ **Figure 3** General structure of a region of the (n, m) -escalator. The rectangles represent the (p, q) -grained gadgets \mathcal{H}_i^j .

We first describe the gadgets related to the vertices. Please refer to Figure 3 to follow the construction. The values of p, q used next will be defined later. An (n, m) -escalator is an interval model \mathcal{D} formed by $m + 1$ (p, q) -grained gadgets for each v_i , denoted by $\mathcal{H}_i^1, \dots, \mathcal{H}_i^{m+1}$, together with $2m$ link intervals, L_i^1, \dots, L_i^{2m} , such that L_i^{2j-1} and L_i^{2j} weakly intersect \mathcal{H}_i^j to the right and weakly intersect \mathcal{H}_i^{j+1} to the left. Additionally, all the grained gadgets are mutually disjoint, and given $j \in \{1, \dots, m + 1\}$ and $i, i' \in \{1, \dots, n\}$ with $i < i'$, the grained gadget \mathcal{H}_i^j occurs to the left of $\mathcal{H}_{i'}^j$.

Now, we add the gadgets related to the edges. Please refer to Figure 4 to follow the construction. The values of p', q' used next will be defined later. For each edge $e_j = v_i v_{i'} \in E(G)$, with $i < i'$, create a (p', q') -grained gadget \mathcal{E}_j and intervals $C_j^1, C_j^2, C_j^3, C_j^4$ in such a way that \mathcal{E}_j is entirely contained in the j -th region (i.e., in the open interval between

the right endpoint of \mathcal{H}_n^j and the left endpoint of \mathcal{H}_1^{j+1} , C_j^1 and C_j^2 weakly intersect \mathcal{H}_i^j to the right and weakly intersect \mathcal{E}_j to the left, and C_j^3 and C_j^4 weakly intersect $\mathcal{H}_{i'}^j$ to the right and strongly intersect \mathcal{E}_j to the left. Denote the constructed model by $\mathcal{M}(\mathfrak{G})$.



■ **Figure 4** General structure of the constructed interval model $\mathcal{M}(\mathfrak{G})$ highlighting the intersections between the intervals of the (n, m) -escalator \mathcal{D} , the intervals of the (p, q') -grained gadget \mathcal{E}_j , and the intervals $C_j^1, C_j^2, C_j^3, C_j^4$.

2.3 Outline of the proof

As above, consider a cubic graph G on n vertices and $m = \frac{3n}{2}$ edges, and let $\pi_V = (v_1, \dots, v_n)$ be an ordering of $V(G)$, $\pi_E = (e_1, \dots, e_m)$ be an ordering of $E(G)$ and $\mathfrak{G} = (G, \pi_V, \pi_E)$. We give an outline of the proof that $\text{mc}(G) \geq k$ if and only if $\text{mc}(\mathbb{G}_{\mathcal{M}(\mathfrak{G})}) \geq f(G, k)$, where f is defined at the end of the subsection. As it is usually the case in this kind of reduction, constructing an appropriate cut of the reduction graph $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$, given a cut of G , is an easy task. On the other hand, constructing an appropriate cut $[X, Y]$ of G , from a given a cut $[A, B]$ of the reduction graph $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$, requires that the intervals in $\mathcal{M}(\mathfrak{G})$ behave in a way with respect to $[A, B]$ so that $[X, Y]$ can be inferred, a task achieved with the help of Lemmas 2 and 3. In order to use these lemmas, we choose next suitable values for p, q, p', q' , and we observe that $\mathcal{M}(\mathfrak{G})$ respects the structure of the involved grained gadgets. After ensuring that each grained gadget behaves well individually, we also need to ensure that \mathcal{H}_i^1 can be used to decide in which part of $[X, Y]$ we should put v_i , and for this it is necessary that all gadgets related to v_i agree with one another. In other words, for each v_i , we want that the behaviour of the first gadget \mathcal{H}_i^1 influence the behaviour of the subsequent gadgets $\mathcal{H}_i^2, \dots, \mathcal{H}_i^{m+1}$, as well as the behaviour of the gadgets related to edges incident to v_i . This is done by choosing the following values for our floating variables:

$$q = 60n^3 + 1, \quad p = 2q + 7n, \quad q' = 18n^3 + 1 \quad \text{and} \quad p' = 2q' + 5n. \quad (2)$$

These values indeed satisfy Conditions (1) and (2) of Lemma 2, as well as Inequality (1). As previously said, the idea behind this choice of values is to store information about v_i in the gadgets $\mathcal{H}_i^1, \dots, \mathcal{H}_i^{m+1}$. Now, given $e_j = v_i v_{i'}$, $i < i'$, a final ingredient is to ensure that \mathcal{E}_j is influenced only by the intervals C_j^3 and C_j^4 , which in turn are influenced by the vertex $v_{i'}$, in a way that the number of edges in the cut-set of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$ increases when the edge $v_i v_{i'}$ is in the cut-set of G . All these ideas are captured in the definitions below.

Given $v_i \in V(G)$ and a cut $[A, B]$ of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$, we say that *the gadgets of v_i alternate in $[A, B]$* if, for every $j \in [m]$, we get that \mathcal{H}_i^j is A -partitioned if and only if \mathcal{H}_i^{j+1} is B -partitioned. Also, we say that $[A, B]$ is *alternating partitioned* if the gadgets of v_i alternate in $[A, B]$, for every $v_i \in V(G)$, and the following conditions hold for every $e_j = v_i v_{i'} \in E(G)$, with $i < i'$:

- (i) If \mathcal{H}_i^j is A -partitioned by $[A, B]$, then $\{C_j^1, C_j^2\} \subseteq B$; otherwise, $\{C_j^1, C_j^2\} \subseteq A$; and
- (ii) If $\mathcal{H}_{i'}^j$ is A -partitioned by $[A, B]$, then $\{C_j^3, C_j^4\} \subseteq B$ and \mathcal{E}_j is A -partitioned by $[A, B]$; otherwise, $\{C_j^3, C_j^4\} \subseteq A$ and \mathcal{E}_j is B -partitioned by $[A, B]$.

The following lemma is a key element in our proof.

► **Lemma 4.** *If $[A, B]$ is a maximum cut of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$, then $[A, B]$ is an alternating partitioned cut.*

Sketch. The proof of Conditions (i) and (ii) is similar to the proof of Lemma 3, and in fact the same ideas are also part of the proof in [1]. Our ability to bound the interval count is due mainly to the fact that the vertex gadgets alternate in $[A, B]$, so we focus on this part of the proof. Another skipped detail is the fact that the pairs of link intervals, and the pairs of *intervals of type C* always go together. More formally, for every $j \in \{1, \dots, m\}$, we have that C_j^1, C_j^2 are in the same part, as well as C_j^3, C_j^4 . Similarly, for every $j \in \{1, \dots, m\}$ and $i \in \{1, \dots, n\}$, the intervals L_i^{2j-1}, L_i^{2j} are in the same part. Just to give an idea for the latter types of intervals, this is due to the fact that the intervals in \mathcal{H}_i^j or \mathcal{H}_{i+1}^j outweigh the total number of *relevant* intervals intersecting L_i^{2j-1}, L_i^{2j} that are outside such vertex gadgets.

Denote $\mathcal{M}(\mathfrak{G})$ by \mathcal{M} for simplicity, and let \mathcal{M}_i be the set of all the intervals related to vertex v_i ; more formally, it contains the grained gadget \mathcal{H}_i^j , for every $j \in [m+1]$, the link interval L_i^j , for every $j \in \{1, \dots, 2m\}$, every interval of type C_j^h that intersects \mathcal{H}_i^j to the right (this happens if e_j has v_i as endpoint), and every interval in \mathcal{E}_j for e_j incident to v_i . We count the number f_i of edges of the cut incident to some interval in \mathcal{M}_i and argue that, if the gadgets of v_i do not alternate in $[A, B]$, then we can obtain a bigger cut by rearranging \mathcal{M}_i , thus getting a contradiction.

Denote by $\overline{\mathcal{M}}_i$ the set of intervals $\mathcal{M} \setminus \mathcal{M}_i$, and by \mathcal{L} the set of all link intervals. In what follows, we do the counting in terms of m, n, p, q, p', q' for simplicity, and we do not make an exact counting, since it would be tedious and not help so much in the understanding of the ideas behind the proof. Also, there will be some values that should be added to f_i that remain the same, independently from how \mathcal{M}_i is partitioned; we call these values *irrelevant* and do not add them to f_i . Recall that every (x, y) -grained gadget has exactly $x + y$ intervals in A and $x + y$ in B . Thus, for each $j \in \{1, \dots, 2m\}$, we know that the number of edges between L_i^j and intervals in $\overline{\mathcal{M}}_i$ that are within a grained gadget do not change if we switch L_i^j from A to B or vice-versa; in other words, these values are irrelevant. Additionally, because we are considering that Conditions (i) and (ii) hold, the number of edges of the cut within each grained gadget of \mathcal{M}_i , and between grained gadgets of type \mathcal{H}_i^j and intervals of \mathcal{M}_i of type C can also be considered irrelevant. So now, for each $j \in \{1, \dots, m\}$, denote by ℓ_A^j the number of intervals in $\overline{\mathcal{M}}_i \cap \mathcal{L} \cap A$ that intersect L_i^{2j} ; define ℓ_B^j similarly. Observe that $\ell_A^j + \ell_B^j \leq 4n$ since it includes all the link intervals in the j -th region, plus at most the link intervals of the $(j-1)$ -th region related to $v_{i'}$ for $i' > i$, and the link intervals of the $(j+1)$ -th region related to $v_{i'}$ for $i' < i$. Additionally, let a_j be equal to 1 if L_i^{2j} is opposite to the right long intervals of \mathcal{H}_i^j , and 0 otherwise; similarly, let b_j be equal to 1 if L_i^{2j} is opposite to the left long intervals of \mathcal{H}_i^{j+1} , and 0 otherwise. Now, let $e_{j_1}, e_{j_2}, e_{j_3}$ be the edges incident to v_i , and for each $h \in \{1, 2, 3\}$, write e_{j_h} as $v_i v_{i_h}$. For each $h \in \{1, 2, 3\}$,

observe that the non-irrelevant number of edges of the cut incident to \mathcal{E}_{j_h} is $2(p' + q')$ if \mathcal{H}_i^j and $\mathcal{H}_{i_h}^j$ are partitioned differently, and that it is equal to $2p'$ otherwise. Therefore, if we let c_h be equal to 1 if \mathcal{H}_i^j and $\mathcal{H}_{i_h}^j$ are partitioned differently, and 0 otherwise, we get that there are $2p' + 2q'c_h$ edges in the cut incident to \mathcal{E}_{j_h} . Because $2p'$ is added for each $h \in \{1, 2, 3\}$, there is an irrelevant value of $6p'$ that we ignore. Therefore, we get that (recall that L^{2j-1} and L^{2j} are true twins):

$$f_i \leq \sum_{j=1}^m (2q(a_j + b_j) + \ell_A^j + \ell_B^j) + \sum_{h=1}^3 2q'c_h. \quad (3)$$

If L_i^{2j} is on the same side as the right long intervals of \mathcal{H}_i^j and the left long intervals of \mathcal{H}_i^{j+1} , we can increase f_i simply by switching its side (together of course with L_i^{2j-1}). Indeed, in this case we would lose at most $\max\{\ell_A^j, \ell_B^j\} \leq 4n$ edges, while gaining $4q$, a positive exchange since $q > n$. Observe that this implies that $a_j + b_j \geq 1$. Note also that this type of argument can be always applied, i.e., whenever in what follows we switch sizes of some subset of intervals, we can suppose that this property still holds. Now, let j be the minimum value for which $a_j + b_j = 1$ (j is well defined since otherwise we get that the gadgets of v_i alternate in $[A, B]$ and there is nothing to prove). Observe that this means that either both \mathcal{H}_i^j and \mathcal{H}_i^{j+1} are A -partitioned, or both are B -partitioned. Suppose the former, without loss of generality, and note that this means that $\mathcal{RL}(\mathcal{H}_i^j) \subseteq A$, while $\mathcal{LL}(\mathcal{H}_i^{j+1}) \subseteq B$. Also, let $j' > j$ be the minimum value for which the left long intervals of $\mathcal{H}_i^{j'+1}$ are on the opposite side of the right long intervals of $\mathcal{H}_i^{j'}$; if it does not exist, let $j' = m + 1$. We switch sides of the following intervals: \mathcal{H}_i^h , for every $h \in \{j + 1, \dots, j'\}$; L_i^{2j-1}, L_i^{2j} if they are also in A ; L_i^{2h-1}, L_i^{2h} for each $h \in \{j + 1, \dots, j' - 1\}$; and $L_i^{2j'-1}, L_i^{2j'}$ if $j' < m + 1$ and they are on the same side as $\mathcal{LL}(\mathcal{H}_i^{j'+1})$. Also switch the intervals of type C and intervals in edge gadgets appropriately in order to maintain the desired properties. We prove that we gain at least $2q - 4n$ edges, while losing at most $4nm + 6q' = 6(n^2 + q')$ (recall that $m = \frac{3n}{2}$). As previously said, this is not the exact count but gives an idea as how to choose the values for p, q, p', q' . Indeed, it suffices to choose values in a way as to ensure that the number of gained edges is bigger than the number of lost edges.

Observe that if we did not need to switch L_i^{2j-1}, L_i^{2j} , then, concerning these intervals, we gain at least $2q$ edges and lose none; otherwise, we gain $2q$ edges but lose at most $\ell_B^j \leq 4n$; thus we gain at least $2q - 4n$. As for the intervals L_i^{2h-1}, L_i^{2h} for $h \in \{j + 1, \dots, j' - 1\}$, by the definition of j' we know that we lose at most $\max\{\ell_A^h, \ell_B^h\} \leq 4n$, while maintaining the same number between them and the vertex gadgets. And if $j' < m + 1$, then we either gain $2q$ more edges if we did not need to change the side of $L_i^{2j'-1}, L_i^{2j'}$, or we gain $2q$ more edges while losing at most $\max\{\ell_A^{j'}, \ell_B^{j'}\} \leq 4n$. Hence, concerning the link intervals in \mathcal{M}_i , in total we lose at most $4nm = 6n^2$. As for the $6q'$ value, it suffices to see that, in the worst case scenario, $\{j_1, j_2, j_3\} \subseteq \{j + 1, \dots, j'\}$ and all the values c_h were previously equal to 1, and are now equal to 0 (observe again Inequality 3). \blacktriangleleft

Now, if $[A, B]$ is an alternating partitioned cut of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$, we let $\Phi(A, B) = [X, Y]$ be the cut of G such that, for each vertex $v_i \in V(G)$, we have $v_i \in X$ if and only if \mathcal{H}_i^1 is A -partitioned by $[A, B]$. Note that $[X, Y]$ is well-defined and uniquely determined by $[A, B]$. On the other hand, given a cut $[X, Y]$ of G , there is a unique alternating partitioned cut $[A, B] = \Phi^{-1}(X, Y)$ of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$ such that $[X, Y] = \Phi(A, B)$. Therefore, it remains to relate the sizes of these cut-sets. Basically we can use the good behaviour of the cuts in $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$ to prove that the size of $[A, B]$ grows as a well-defined function on the size of $\Phi(A, B)$. More formally, we can prove that the function f previously referred to is given by (recall that k is part of the input on the original problem):

$$f(G, k) = \left(\frac{3n^2}{2} + n\right)(2pq + q^2) + \frac{3n}{2}(2p'q' + (q')^2) + 6nq(n+1) \\ + (3n^2 + 3n)(n-1)(p+q) + 3n^2(p'+q') + 3n((k+1)q' + p') + 4k. \quad (4)$$

The above value is obtained using counting arguments much similar to the ones given in the proof of Lemma 4. The only downside is that we are not able to give an exact value for $|E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)|$ as a function of $|E_G(X, Y)|$ and n, p, q, p', q' . For instance, note that the number of edges between link intervals within a region depend on the size of A and B , instead of the size of $E_G(X, Y)$. Nevertheless, we know that the range of values that $|E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)|$ can assume, given that $|E_G(X, Y)| = k$, is distinct for each value of k , as the following lemma states.

► **Lemma 5.** *Let G be a cubic graph on n vertices, $\pi_V = (v_1, \dots, v_n)$ be an ordering of $V(G)$, $\pi_E = (e_1, \dots, e_{\frac{3n}{2}})$ be an ordering of $E(G)$, $\mathfrak{G} = (G, \pi_V, \pi_E)$, $[A, B]$ be an alternating partitioned cut of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$ and $[X, Y] = \Phi(A, B)$. If $k = |E_G(X, Y)|$, then $f(G, k) \leq |E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)| < f(G, k')$ for any integer $k' > k$.*

Sketch. Since $[A, B]$ is an alternating partitioned cut of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$, we shall count the edges in the cut-set $E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)$ according to the following three types of intervals incident to these edges: the edges in the cut-set that have an endpoint in a (p, q) -grained gadget; the edges in the cut-set that have an endpoint in a (p', q') -grained gadget; and the edges in the cut-set that have both endpoints in a link interval and/or an interval of the type C_j^ℓ .

First, we count the edges in the cut-set that have an endpoint in a (p, q) -grained gadget. The possible combinations are as follows.

- (1.1) Edges within (p, q) -grained gadgets related to vertices. There are exactly $(\frac{3n^2}{2} + n)(2pq + q^2)$ such edges.
- (1.2) Edges between link intervals L_i^{2j-1} and L_i^{2j} , and the (p, q) -gadgets related to vertices. There are exactly $m \cdot n \cdot (2q + 2q) = 6n^2q$ such edges.
- (1.3) Edges between intervals C_j^1, \dots, C_j^4 and the (p, q) -grained related to the vertices incident to edge e_j . There are exactly $\frac{3n}{2}(2q + 2q) = 6nq$ such edges.
- (1.4) Edges between (p, q) -grained gadgets related to vertices, and link intervals covering them. There are exactly $mn(n-1)(2p + 2q) = 3n^2(n-1)(p+q)$ such edges.
- (1.5) Edges between intervals C_j^1, \dots, C_j^4 and (p, q) -grained gadgets covered by them. There are exactly $\sum_{i \in [n]} 6(n-i)(p+q) = 3n(n-1)(p+q)$ such edges.

Second, we count the edges in the cut-set that have an endpoint in a (p', q') -grained gadget. The possible combinations are as follows.

- (2.1) Edges within (p', q') -grained gadgets related to edges. There are exactly $\frac{3n}{2}(2p'q' + (q')^2)$ such edges.
- (2.2) Edges between (p', q') -grained gadgets related to edges and the link intervals covering them. There are exactly $3n^2(p' + q')$ such edges.
- (2.3) Edges between (p', q') -grained gadget \mathcal{E}_j and intervals C_j^1, \dots, C_j^4 . There are exactly $\frac{3n}{2}(2kq' + 2(p' + q')) = 3n((k+1)q' + p')$ such edges (recall that $k = |E_G(X, Y)|$).

Third, we count the edges in the cut-set that have both endpoints in a link interval and/or an interval of the type C_j^ℓ for some $\ell \in \{1, \dots, 4\}$ and $j \in [m]$.

38:10 Maximum Cut on Interval Graphs of Interval Count Four Is NP-Complete

- (3.1) Edges between intervals C_j^1, C_j^2 and C_j^3, C_j^4 . There are exactly $4k$ such edges.
- (3.2) Edges between pairs of intervals $L_1^{2j-1}, L_1^{2j}, \dots, L_n^{2j-1}, L_n^{2j}$. There are *at most* $\sum_{j \in [m]} n^2 = mn^2 = \frac{3n^3}{2}$ such edges.
- (3.3) Edges between intervals $L_1^{2j-1}, L_1^{2j}, \dots, L_n^{2j-1}, L_n^{2j}$ and intervals in C_j^1, \dots, C_j^4 . There are *at most* $\sum_{j \in [m]} 8(n-1) = 8m(n-1) = 12n(n-1) = 12n^2 - 12n$ such edges.
- (3.4) Edges between link intervals in consecutive regions of the escalator. There are *at most*

$$\begin{aligned} \sum_{j \in \{2, \dots, m\}} \sum_{i \in [n]} 4(n-i) &= \sum_{j \in \{2, \dots, m\}} 2n(n-1) = 2(m-1)n(n-1) \\ &= 3n^2(n-1) - 2n(n-1) = 3n^3 - 5n^2 + 2n \end{aligned}$$

such edges.

- (3.5) Finally, edges between intervals C_j^1, \dots, C_j^4 and link intervals in the previous regions of the escalator. There are *at most* $\sum_{i \in [n]} 12(n-i) = 6n^2 - 6n$ such edges.

Therefore, summing up the number of edges in the cut-set $E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)$ according to three types described above, except for the edges described in Cases (3.2)–(3.5) which, as we have seen, do not give exact values, we obtain that

$$\begin{aligned} |E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)| &\geq \left(\frac{3n^2}{2} + n\right) (2pq + q^2) + \frac{3n}{2} (2p'q' + (q')^2) + 6nq(n+1) \\ &\quad + (3n^2 + 3n)(n-1)(p+q) + 3n^2(p'+q') \\ &\quad + 3n((k+1)q' + p') + 4k \\ &= f(G, k). \end{aligned}$$

On the other hand, note that the number of edges in Cases (3.2)–(3.5) is upper bounded by $\frac{9n^3}{2} + 13n^2 - 16n$. Thus, since $q' > \frac{9n^3}{2} + 13n^2 - 16n$, we have:

$$f(G, k) \leq |E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)| \leq f(G, k) + \frac{9n^3}{2} + 13n^2 - 16n < f(G, k) + q'.$$

As a result, because there is a factor kq' in $f(G, k)$, we obtain that $f(G, k') > |E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)|$ for any $k' > k$. \blacktriangleleft

The next lemma together with Lemma 7 stated next in Section 2.4 complete the proof of Theorem 1.

► **Lemma 6.** *Let G be a cubic graph on n vertices, $\pi_V = (v_1, \dots, v_n)$ be an ordering of $V(G)$, $\pi_E = (e_1, \dots, e_{3n})$ be an ordering of $E(G)$ and $\mathfrak{G} = (G, \pi_V, \pi_E)$. For each positive integer k , $\text{mc}(G) \geq k$ if and only if $\text{mc}(\mathbb{G}_{\mathcal{M}(\mathfrak{G})}) \geq f(G, k)$.*

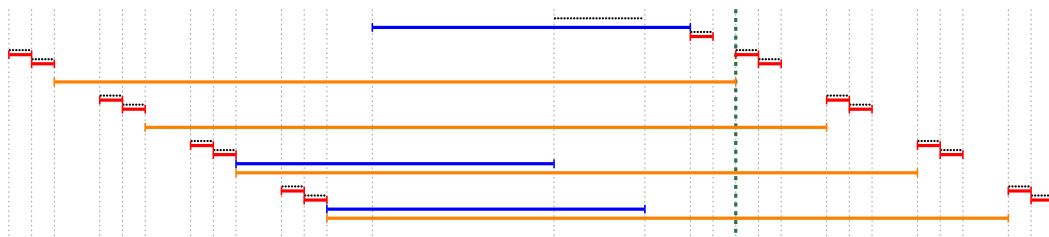
Proof. First, suppose that $\text{mc}(G) \geq k$. Then, there is a cut $[X, Y]$ of G such that $|E_G(X, Y)| \geq k$. Let $[A, B]$ be the unique alternating partitioned cut of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$ that, for each $i \in [n]$, satisfies the following condition: if $v_i \in X$, then \mathcal{H}_i^1 is A -partitioned; otherwise, \mathcal{H}_i^1 is B -partitioned. One can verify that $[A, B] = \Phi^{-1}(X, Y)$. Therefore, it follows from Lemma 5 that $\text{mc}(\mathbb{G}_{\mathcal{M}(\mathfrak{G})}) \geq |E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)| \geq f(G, k)$. Conversely, suppose that $\text{mc}(\mathbb{G}_{\mathcal{M}(\mathfrak{G})}) \geq f(G, k)$. Then, there exists a cut $[A, B]$ of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$ such that $|E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)| \geq f(G, k)$. Assume that $[A, B]$ is a maximum cut of $\mathbb{G}_{\mathcal{M}(\mathfrak{G})}$. It follows from Lemma 4 that $[A, B]$ is an alternating partitioned cut. Consequently, by Lemma 5, $[X, Y] = \Phi(A, B)$ is a cut of G such that $|E_G(X, Y)| \geq k$. Indeed, if $|E_G(X, Y)| < k$, then $|E_{\mathbb{G}_{\mathcal{M}(\mathfrak{G})}}(A, B)| < f(G, k)$. Therefore, $\text{mc}(G) \geq k$. \blacktriangleleft

2.4 Bounding the interval count

Consider a cubic graph G on n vertices and $m = \frac{3n}{2}$ edges, and orderings π_V, π_E of the vertex set and edge set of G . Denote the triple (G, π_V, π_E) by \mathfrak{G} . We want to prove that the interval count of our constructed interval model $\mathcal{M}(\mathfrak{G})$ is at most 4. But observe that the construction of $\mathcal{M}(\mathfrak{G})$ is actually not unique, since the intervals are not uniquely defined; e.g., given such a model, one can obtain a model satisfying the same properties simply by adding $\epsilon > 0$ to all points defining the intervals. In this section, we provide a construction of a uniquely defined interval model related to \mathfrak{G} that satisfies the desired conditions and has interval count 4.

Consider our constructed interval model $\mathcal{M}(\mathfrak{G})$, and denote, for each $j \in [m]$, $\mathcal{S}_j = \mathcal{E}_j \cup \bigcup_{\ell \in [4]} C_j^\ell \cup \bigcup_{i \in [n]} (\mathcal{H}_i^j \cup \{L_i^{2j} \cup L_i^{2j-1}\})$. We show how to accommodate \mathcal{S}_1 within $[0, 6n - 2]$ in such a way that the same pattern can be adopted in the subsequent regions of $\mathcal{M}(\mathfrak{G})$ too, each time starting at multiples of $4n$. More specifically, letting $t = 4n$, we will accommodate \mathcal{S}_j within $[t \cdot (j - 1), 6n - 2 + t \cdot (j - 1)]$. Assume $e_1 = v_h v_{h'}$, with $h < h'$. Below, we say exactly which closed interval of the line corresponds to each interval $I \in \mathcal{S}_1$.

- For each $i \in [n]$, the left long intervals of \mathcal{H}_i^1 are equal to $[2i - 2, 2i - \frac{3}{2}]$ and the left short intervals are any choice of q distinct points within the open interval $(2i - 2, 2i - \frac{3}{2})$, whereas the right long intervals of \mathcal{H}_i^1 are equal to $[2i - \frac{3}{2}, 2i - 1]$ and the right short intervals are any choice of q distinct points within the open interval $(2i - \frac{3}{2}, 2i - 1)$. Note that open intervals are used to locate the closed intervals of length zero, but that the short intervals themselves are not open.
- C_1^1 and C_1^2 are equal to $[2h - 1, 2h + 2n - 2]$.
- C_1^3 and C_1^4 are equal to $[2h' - 1, 2h' + 2n - 2]$.
- The left long intervals of \mathcal{E}_1 are equal to $[2n, 4n - 1]$.
- The left short intervals of \mathcal{E}_1 are any choice of q' distinct points in the open interval $(2h + 2n - 2, 2h' + 2n - 2)$. Again, the open interval is used just to locate the closed intervals of length zero.
- The right long intervals of \mathcal{E}_1 are equal to $[4n - 1, 4n - \frac{1}{2}]$ and the right short intervals are any choice of q' distinct points within the corresponding open interval.
- For each $i \in [n]$, intervals L_i^1, L_i^2 are equal to $[2i - 1, 4n + 2(i - 1)]$.



■ **Figure 5** The closed intervals in $\mathcal{S}_1 \cup \bigcup_{i=1}^4 \mathcal{H}_i^2$ of a graph on 4 vertices. We consider e_1 to be equal to $v_3 v_4$. Each colour represents a different interval size. The short intervals are represented by the dots located inside the open interval. Vertical lines mark the endpoints of the intervals in $\mathcal{S}_1 \setminus \mathcal{L}$, while the green vertical line marks the beginning of the intervals in \mathcal{S}_2 .

The suitable chosen lengths of the above defined closed intervals are (see Figure 5, where we denote by \mathcal{L} the set of link intervals):

1. 0: short intervals of all grained gadgets (dots in Figure 5);
2. $1/2$: left long and right long intervals of each \mathcal{H}_i^1 , and right long intervals of \mathcal{E}_1 (red intervals in Figure 5);
3. $2n - 1$: intervals C_1^1, \dots, C_1^4 , and left long intervals of \mathcal{E}_1 (blue intervals in Figure 5);
4. $4n - 1$: intervals L_i^1 and L_i^2 , for every $i \in [n]$ (orange intervals in Figure 5).

Now, let $\mathcal{M}'(\mathfrak{G})$ be the interval model where each \mathcal{S}_j is defined exactly as \mathcal{S}_1 , except that we shift all the intervals to the right in a way that point 0 now coincides with point $t \cdot (j - 1)$. More formally, an interval I in \mathcal{S}_j corresponding to the copy of an interval $[\ell, r]$ in \mathcal{S}_1 is defined as $[\ell + t \cdot (j - 1), r + t \cdot (j - 1)]$. Also, we assign the intervals in the $(m + 1)$ -th grained gadgets to be at the end of this model, using the same sizes of intervals as above; i.e., \mathcal{H}_i^{m+1} is within the interval $[2i - 2 + t \cdot m, 2i - 1 + t \cdot m]$.

We have shown above that $\mathcal{M}'(\mathfrak{G})$ has interval count 4. The following lemma shows that the above chosen intervals satisfy the properties imposed in Subsections 2.1 and 2.2 on our constructed interval model $\mathcal{M}(\mathfrak{G})$.

► **Lemma 7.** *Let G be a cubic graph. Then, there exists an interval model $\mathcal{M}(\mathfrak{G})$ with interval count 4 for $\mathfrak{G} = (G, \pi_V, \pi_E)$, for every ordering π_V and π_E of the vertex set and edge set of G , respectively.*

3 The interval count of Adhikary et al.'s construction

We provided in Section 2 a reduction from the MAXCUT problem having as input a cubic graph G into that of MAXCUT in an interval graph G' having $\text{ic}(G') \leq 4$. Although our reduction requires the choice of orderings π_V and π_E of respectively $V(G)$ and $E(G)$ in order to produce the resulting interval model, we have established that we are able to construct an interval model with interval count 4 regardless of the particular choices for π_V and π_E (Lemma 7). Our reduction was based on that of [1], strengthened in order to control the interval count of the resulting model.

This section is dedicated to discuss the interval count of the original reduction [1]. Although the interval count was not of concern in [1], in order to contrast the reduction found there with the presented in this work, we investigate how interval count varies in the original reduction considering different vertex/edge orderings. First, we establish that the original reduction yields an interval model corresponding to a graph G' such that $\text{ic}(G') = O(\sqrt[4]{|V(G')|})$. Second, we exhibit an example of a cubic graph G for which a choice of π_V and π_E yields a model \mathcal{M}' with interval count $\Omega(\sqrt[4]{|V(G')|})$, proving that this bound is tight for some choices of π_V and π_E . For bridgeless cubic graphs, we are able in Lemma 8 to decrease the upper bound by a constant factor, but to the best of our knowledge $O(\sqrt[4]{|V(G')|})$ is the tightest upper bound. Before we go further analysing the interval count of the original reduction, it is worthy to note that a tight bound on the interval count of a general interval graph G as a function of its number of vertices n is still open. It is known that $\text{ic}(G) \leq \lfloor (n + 1)/2 \rfloor$ and that there is a family of graphs G for which $\text{ic}(G) = (n - 1)/3$ [7, 14]. That is, the interval count of a graph can achieve $\Theta(n)$.

In the original reduction, given a cubic graph G , an interval graph G' is defined through the construction of one of its models \mathcal{M} , described as follows:

1. let $\pi_V = (v_1, v_2, \dots, v_n)$ and $\pi_E = (e_1, e_2, \dots, e_m)$ be arbitrary orderings of $V(G)$ and $E(G)$, respectively;
2. for each $v_i \in V(G)$, $e_j \in E(G)$, let $\mathcal{G}(v_i)$ and $\mathcal{G}(e_j)$ denote respectively a (p, q) -grained gadget and a (p', q') -grained gadget, where:
 - $q = 200n^3 + 1$, $p = 2q + 7n$, and
 - $q' = 10n^2 + 1$, $p' = 2q' + 7n$;

3. for each $v_k \in V(G)$, insert $\mathcal{G}(v_k)$ in \mathcal{M} such that $\mathcal{G}(v_i)$ is entirely to the left of $\mathcal{G}(v_j)$ if and only if $i < j$. For each $e_k \in E(G)$, insert $\mathcal{G}(e_k)$ in \mathcal{M} entirely to the right of $\mathcal{G}(v_n)$ and such that $\mathcal{G}(e_i)$ is entirely to the left of $\mathcal{G}(e_j)$ if and only if $i < j$;
4. for each $e_j = (v_i, v_{i'}) \in E(G)$, with $i < i'$, four intervals $I_{i,j}^1, I_{i,j}^2, I_{i',j}^1, I_{i',j}^2$ are defined in \mathcal{M} , called *link* intervals, such that:
 - $I_{i,j}^1$ and $I_{i,j}^2$ (resp. $I_{i',j}^1$ and $I_{i',j}^2$) are true twin intervals that weakly intersect $\mathcal{G}(v_i)$ (resp. $\mathcal{G}(v_{i'})$) to the right;
 - $I_{i,j}^1$ and $I_{i',j}^1$ (resp. $I_{i,j}^2$ and $I_{i',j}^2$) weakly intersect (resp. strongly intersect) $\mathcal{G}(e_j)$ to the left.

By construction, therefore, $I_{i,j}^1$ and $I_{i,j}^2$ (resp. $I_{i',j}^1$ and $I_{i',j}^2$) cover all intervals in grained gadgets associated to a vertex v_ℓ with $\ell > i$ (resp. $\ell > i'$) or an edge e_ℓ with $\ell < j$.

Note that the number of intervals in \mathcal{M} is invariant under the particular choices of π_V and π_E and, therefore, so is the number of vertices of G' . Let $n' = |V(G')|$. Since G is cubic, $m = \frac{3n}{2}$. By construction,

$$n' = n(2p + 2q) + m(2p' + 2q') + 4m = 1200n^4 + 90n^3 + 25n^2 + 21n$$

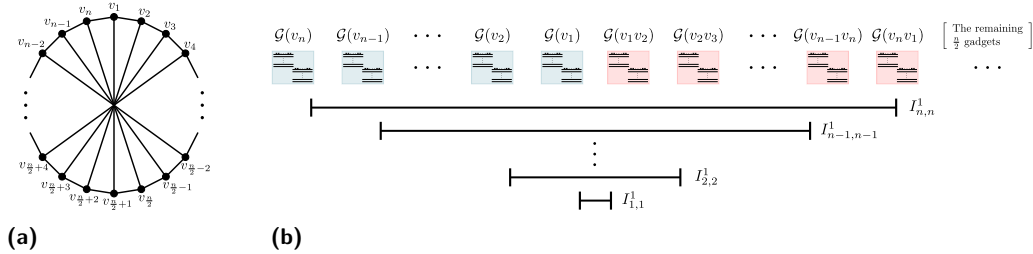
and thus $n = \Theta(\sqrt[4]{n'})$. Since the set of intervals covered by any link interval depends on π_V and π_E , distinct sequences yield distinct resulting graphs G' having distinct interval counts.

We show next that $\text{ic}(G') = O(\sqrt[4]{n'})$. Note that

- the intervals of all gadgets $\mathcal{G}(v_i)$ and $\mathcal{G}(e_j)$ can use only two interval lengths (one for all short intervals, another for all the long intervals);
- for each $e_j = v_i v_{i'} \in E(G)$, with $i < i'$, both intervals $I_{i,j}^1$ and $I_{i,j}^2$ may be coincident in any model, and therefore may have the same length. The same holds for both intervals $I_{i',j}^1$ and $I_{i',j}^2$.

Therefore, $\text{ic}(G') \leq 2m + 2 = 3n + 2 = \Theta(\sqrt[4]{n'})$. Therefore, the NP-completeness result derived from the original reduction in [1] can be strengthened to state that MAXCUT is NP-complete for interval graphs G having interval count $O(\sqrt[4]{|V(G)|})$.

Second, we show that there is a resulting model \mathcal{M}' produced in the reduction, defined in terms of particular orderings π_V, π_E for which $\text{ic}(\mathcal{M}') = \Omega(\sqrt[4]{n'})$. Consider the cubic graph G depicted in Figure 6(a) which consists of an even cycle (v_1, v_2, \dots, v_n) with the addition of the edges $(v_i, v_{i+\frac{n}{2}})$ for all $1 \leq i \leq n/2$. For the ordering $\pi_V = (v_n, v_{n-1}, \dots, v_1)$ and any ordering π_E in which the first n edges are the edges of the cycle (v_1, v_2, \dots, v_n) , in this order, the reduction yields a model \mathcal{M}' for which there is a chain $I_{1,1}^1 \subset I_{2,2}^1 \subset \dots \subset I_{n,n}^1$ of nested intervals (see Figure 6(b)), which shows that $\text{ic}(\mathcal{M}') \geq n$, and thus $\text{ic}(\mathcal{M}') = \Omega(\sqrt[4]{n'})$.



■ **Figure 6** (a) A cubic graph G , and (b) a chain of nested intervals in the model \mathcal{M}' .

It can be argued from the proof of NP-completeness for MAXCUT when restricted to cubic graphs [2] that the constructed cubic graph may be assumed to have no bridges. This fact was not used in the original reduction of [1]. In an attempt to obtain a model \mathcal{M} having

fewer lengths for bridgeless cubic graphs, we have derived Lemma 8. Although the number of lengths in this new upper bound has decreased by the constant factor of $4/9$, it is still $\Theta(n) = \Theta(\sqrt[4]{n'})$.

► **Lemma 8.** *Let G be a cubic bridgeless graph with $n = |V(G)|$. There exist particular orderings π_V of $V(G)$ and π_E of $E(G)$ such that:*

1. *there is a resulting model \mathcal{M} produced in the original reduction of MAXCUT such that $\text{ic}(\mathcal{M}) \leq \frac{4n}{3} + 3$.*
2. *for all such resulting models \mathcal{M} , we have that $\text{ic}(\mathcal{M}) \geq 5$ if G is not a Hamiltonian graph.*

As a concluding remark, we note that the interval count of the interval model \mathcal{M} produced in the original reduction is highly dependent on the assumed orderings of $V(G)$ and $E(G)$, and may achieve $\text{ic}(\mathcal{M}) = \Omega(\sqrt[4]{n'})$. The model \mathcal{M}' produced in our reduction enforces that $\text{ic}(\mathcal{M}') = 4$ which is invariant for any such orderings. On the perspective of the problem of interval count 2 and beyond, for which very little is known, our NP-completeness result on a class of bounded interval count graphs is also of interest.

References

- 1 Ranendu Adhikary, Kaustav Bose, Satwik Mukherjee, and Bodhayan Roy. Complexity of maximum cut on interval graphs. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry, SoCG 2021, June 7-11, 2021, Buffalo, NY, USA (Virtual Conference)*, volume 189 of *LIPICs*, pages 7:1–7:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.SoCG.2021.7.
- 2 Piotr Berman and Marek Karpinski. On some tighter inapproximability results (extended abstract). In Jiri Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 1999. doi:10.1007/3-540-48523-6_17.
- 3 Hans L. Bodlaender, Celina M. H. de Figueiredo, Marisa Gutierrez, Ton Kloks, and Rolf Niedermeier. Simple max-cut for split-indifference graphs and graphs with few P_4 's. In Celso C. Ribeiro and Simone L. Martins, editors, *Experimental and Efficient Algorithms, Third International Workshop, WEA 2004, Angra dos Reis, Brazil, May 25-28, 2004, Proceedings*, volume 3059 of *Lecture Notes in Computer Science*, pages 87–99. Springer, 2004. doi:10.1007/978-3-540-24838-5_7.
- 4 Hans L. Bodlaender, Ton Kloks, and Rolf Niedermeier. SIMPLE MAX-CUT for unit interval graphs and graphs with few P_4 s. *Electron. Notes Discret. Math.*, 3:19–26, 1999. doi:10.1016/S1571-0653(05)80014-9.
- 5 J. Adrian Bondy and Uppaluri S. R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2008. doi:10.1007/978-1-84628-970-5.
- 6 Arman Boyaci, Tınaz Ekim, and Mordechai Shalom. A polynomial-time algorithm for the maximum cardinality cut problem in proper interval graphs. *Inf. Process. Lett.*, 121:29–33, 2017. doi:10.1016/j.ipl.2017.01.007.
- 7 Márcia R. Cerioli, Fabiano de S. Oliveira, and Jayme Luiz Szwarcfiter. The interval count of interval graphs and orders: a short survey. *J. Braz. Comput. Soc.*, 18(2):103–112, 2012. doi:10.1007/s13173-011-0047-1.
- 8 Dibyayan Chakraborty, Sandip Das, Florent Foucaud, Harmender Gahlawat, Dimitri Lajou, and Bodhayan Roy. Algorithms and complexity for geodetic sets on planar and chordal graphs. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPICs*, pages 7:1–7:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ISAAC.2020.7.

- 9 Johanne Cohen, Fedor V. Fomin, Pinar Heggernes, Dieter Kratsch, and Gregory Kucherov. Optimal linear arrangement of interval graphs. In Rastislav Kralovic and Pawel Urzyczyn, editors, *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, pages 267–279. Springer, 2006. doi:10.1007/11821069_24.
- 10 Derek G. Corneil, Hiryoung Kim, Sridhar Natarajan, Stephan Olariu, and Alan P. Sprague. Simple linear time recognition of unit interval graphs. *Inf. Process. Lett.*, 55(2):99–104, 1995. doi:10.1016/0020-0190(95)00046-F.
- 11 Celina M. H. de Figueiredo, João Meidanis, and Célia Picinin de Mello. A linear-time algorithm for proper interval graph recognition. *Inf. Process. Lett.*, 56(3):179–184, 1995. doi:10.1016/0020-0190(95)00133-W.
- 12 Celina M.H. de Figueiredo, Alexsander A. de Melo, Diana Sasaki, and Ana Silva. Revising Johnson’s table for the 21st century. *Discret. Appl. Math.*, 2021. doi:10.1016/j.dam.2021.05.021.
- 13 Tınaz Ekim, Aysel Erey, Pinar Heggernes, Pim van ’t Hof, and Daniel Meister. Computing minimum geodetic sets of proper interval graphs. In David Fernández-Baca, editor, *LATIN 2012: Theoretical Informatics – 10th Latin American Symposium, Arequipa, Peru, April 16-20, 2012. Proceedings*, volume 7256 of *Lecture Notes in Computer Science*, pages 279–290. Springer, 2012. doi:10.1007/978-3-642-29344-3_24.
- 14 Peter C. Fishburn. Interval graphs and interval orders. *Discret. Math.*, 55(2):135–149, 1985. doi:10.1016/0012-365X(85)90042-1.
- 15 M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.
- 16 Yuan Jinjiang and Zhou Sanming. Optimal labelling of unit interval graphs. *Applied Math.*, 10(3):337–344, September 1995. doi:10.1007/bf02662875.
- 17 David S. Johnson. The NP-completeness column: An ongoing guide. *J. Algorithms*, 6(3):434–451, 1985. doi:10.1016/0196-6774(85)90012-4.
- 18 Pavel Klavík, Yota Otachi, and Jirí Sejnoha. On the classes of interval graphs of limited nesting and count of lengths. *Algorithmica*, 81(4):1490–1511, 2019. doi:10.1007/s00453-018-0481-y.
- 19 Jan Kratochvíl, Tomáš Masarík, and Jana Novotná. U-bubble model for mixed unit interval graphs and its applications: The maxcut problem revisited. In Javier Esparza and Daniel Král’, editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPICs*, pages 57:1–57:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.MFCS.2020.57.
- 20 Dániel Marx. A short proof of the NP-completeness of minimum sum interval coloring. *Oper. Res. Lett.*, 33(4):382–384, 2005. doi:10.1016/j.orl.2004.07.006.
- 21 Sara Nicoloso, Majid Sarrafzadeh, and X. Song. On the sum coloring problem on interval graphs. *Algorithmica*, 23(2):109–126, 1999. doi:10.1007/PL00009252.

Fuzzy Simultaneous Congruences

Max A. Deppert ✉ 

Kiel University, Germany

Klaus Jansen ✉ 

Kiel University, Germany

Kim-Manuel Klein ✉ 

Kiel University, Germany

Abstract

We introduce a very natural generalization of the well-known problem of simultaneous congruences. Instead of searching for a positive integer s that is specified by n *fixed remainders* modulo integer divisors a_1, \dots, a_n we consider *remainder intervals* R_1, \dots, R_n such that s is feasible if and only if s is congruent to r_i modulo a_i for *some* remainder r_i in interval R_i for all i .

This problem is a special case of a 2-stage integer program with only two variables per constraint which is closely related to directed Diophantine approximation as well as the mixing set problem. We give a hardness result showing that the problem is NP-hard in general.

By investigating the case of *harmonic divisors*, i.e. a_{i+1}/a_i is an integer for all $i < n$, which was heavily studied for the mixing set problem as well, we also answer a recent algorithmic question from the field of real-time systems. We present an algorithm to decide the feasibility of an instance in time $\mathcal{O}(n^2)$ and we show that if it exists even the *smallest* feasible solution can be computed in strongly polynomial time $\mathcal{O}(n^3)$.

2012 ACM Subject Classification Theory of computation → Discrete optimization; Theory of computation → Integer programming

Keywords and phrases Simultaneous congruences, Integer programming, Mixing Set, Real-time scheduling, Diophantine approximation

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.39

Related Version *Full Version*: <https://arxiv.org/abs/2002.07746>

Funding *Max A. Deppert*: Supported by German Research Foundation (DFG) project JA 612/20-1.

Klaus Jansen: Supported by German Research Foundation (DFG) project JA 612/20-1.

1 Introduction

In the recent past there was a great interest in the so-called *n-fold* integer programs [10, 17, 19] and *2-stage* integer programs [18, 20]. The matrix \mathcal{A} of a 2-stage integer program is constructed by block matrices $A^{(1)}, \dots, A^{(n)} \in \mathbb{Z}^{r \times k}$ and $B^{(1)}, \dots, B^{(n)} \in \mathbb{Z}^{r \times t}$ as follows:

$$\mathcal{A} = \begin{pmatrix} A^{(1)} & B^{(1)} & 0 & \dots & 0 \\ A^{(2)} & 0 & B^{(2)} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ A^{(n)} & 0 & \dots & 0 & B^{(n)} \end{pmatrix}$$

For an objective vector $c \in \mathbb{Z}_{\geq 0}^{k+nt}$, a right-hand side $b \in \mathbb{Z}^{nr}$, and bounds $\ell, u \in \mathbb{Z}_{\geq 0}^{k+nt}$ the 2-stage integer program is formulated as

$$\max \{ c^T x \mid \mathcal{A}x = b, \ell \leq x \leq u, x \in \mathbb{Z}^{k+nt} \}.$$

A special case of a 2-stage integer program is given by the problem MIXING SET [6, 7, 15] (with only two variables in each constraint) where especially $r = k = t = 1$ and $A^{(1)} = \dots = A^{(n)}$. Remark that 2-variable integer programming problems were extensively studied by various



© Max A. Deppert, Klaus Jansen, and Kim-Manuel Klein;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 39; pp. 39:1–39:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

39:2 Fuzzy Simultaneous Congruences

authors, e.g. [3, 22] or [12] (with two variables in total). MIXING SET plays an important role for example in integer programming approaches for production planning [26]. Given vectors $a, b \in \mathbb{Q}^n$ one aims to compute

$$\min \{ f(s, x) \mid s + a_i x_i \geq b_i \forall i = 1, \dots, n, (s, x) \in \mathbb{Z}_{\geq 0} \times \mathbb{Z}^n \} \quad (1)$$

for some objective function f . Conforti et al. [8] pose the question whether the problem can be solved in polynomial time for linear functions f . Unless $P = NP$ this was ruled out by Eisenbrand and Rothvoß [13] who proved that optimizing any linear function over MIXING SET is NP-hard. However, the problem can be solved in polynomial time if $a_i = 1$ [15, 24] or if the capacities a_i fulfil a *harmonic* property [30], i.e. a_{i+1}/a_i is integer for all $i < n$. The case of harmonic capacities was intensively studied - see [8, 9] for simpler approaches.

More recently, real-time systems with harmonic tasks (the periods are integer multiples of each other) have received increased attention [5] and also harmonic periods have been considered before [2, 11, 27, 29]. Now a recent manuscript in the field of real-time systems by Nguyen et al. [25] gives rise to the study of a new problem. Nguyen et al. present an algorithm for the worst-case response time analysis of harmonic tasks with constrained release jitter running in polynomial time. The release jitter of a task is the maximum difference between the arrival times and the release times over all jobs of the task. Their algorithm uses heuristic components to solve an integer program that can be stated as a bounded version of MIXING SET with additional upper bounds B_i as follows.

BOUNDED MIXING SET (BMS)

Given capacities $a_1, \dots, a_n \in \mathbb{Z}$ and bounds $b, B \in \mathbb{Z}^n$ find $(s, x) \in \mathbb{Z}_{\geq 0} \times \mathbb{Z}^n$ such that

$$b_i \leq s + a_i x_i \leq B_i \quad \forall i = 1, \dots, n.$$

In particular they depend on minimizing the value of s which can be achieved in linear time in case of MIXING SET. While BMS may look artificial at first sight it is not; in fact, leading to a very natural generalization it can be restated in the well-known form of *simultaneous congruences*.

FUZZY SIMULTANEOUS CONGRUENCES (FSC)

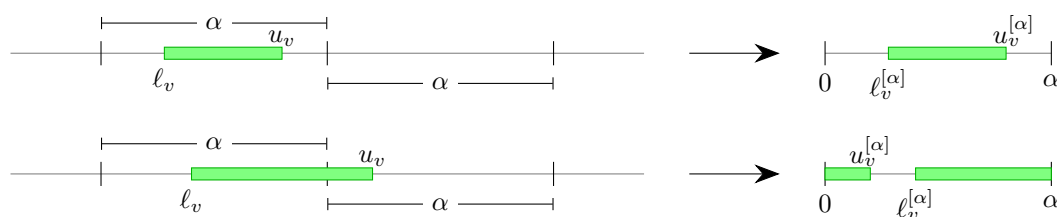
Given divisors $a_1, \dots, a_n \in \mathbb{Z} \setminus \{0\}$ and remainder intervals $R_1, \dots, R_n \subseteq \mathbb{Z}$ and an interval $S \subseteq \mathbb{Z}_{\geq 0}$ find a number $s \in S$ such that

$$\exists r_i \in R_i : s \equiv r_i \pmod{a_i} \quad \forall i = 1, \dots, n.$$

Obviously, this also generalizes over the well-known problem of the Chinese Remainder Theorem (CRT). Here we give its generalized form (cf. [21]).

► **Theorem 1 (Generalized Chinese Remainder Theorem).** *Given divisors $a_1, \dots, a_n \in \mathbb{Z}_{\geq 1}$ and remainders $r_1, \dots, r_n \in \mathbb{Z}_{\geq 0}$ the system of n simultaneous congruences $s \equiv r_i \pmod{a_i}$ admits a solution $s \in \mathbb{Z}$ if and only if $r_i \equiv r_j \pmod{\gcd(a_i, a_j)}$ for all $i \neq j$.*

Furthermore, Leung and Whitehead [23] showed that k -Simultaneous Congruences (k -SC) is NP-complete in the weak sense. Given divisors $a_1, \dots, a_n \in \mathbb{Z}_{\geq 1}$ and remainders $r_1, \dots, r_n \in \mathbb{Z}_{\geq 0}$ the task is to find a number $s \in \mathbb{Z}_{\geq 0}$ and a subset $I \subseteq \{1, \dots, n\}$ with $|I| = k$ s.t. $s \equiv r_i \pmod{a_i}$ for all $i \in I$. Later it was shown by Baruah et al. [4] that k -SC also is NP-complete in the strong sense.



■ **Figure 1** The two possibilities for the modular projection of an interval.

Both problems BMS and FSC are interchangeable formulations of the same problem (see Section 2). Therefore, we will use them as synonyms and we especially assume formally that $R_i = [b_i, B_i]$. Interestingly and to the best of our knowledge, FSC/BMS was not considered before. However, the investigation of simultaneous congruences has always been of transdisciplinary interest connecting a variety of fields and applications, e.g. [1, 14, 16].

Our Contribution

- (a) We show that BMS is NP-hard for general capacities a_i . For the reduction from DIRECTED DIOPHANTINE APPROXIMATION we refer to the appendix. Compared to MIXING SET this is a stronger hardness result as BMS by itself only asks for an *arbitrary* feasible solution. Remark that every feasible instance of MIXING SET may be solved by $s = \|b\|_\infty, x = \mathbf{0}$.
- (b) In the case of harmonic capacities (i.e. a_{i+1}/a_i is an integer for all $i < n$), which was heavily studied for MIXING SET as mentioned before, we give an algorithm exploiting a merge idea based on modular arithmetic on intervals to decide the feasibility problem of FSC in time $\mathcal{O}(n^2)$. See Section 3.1 for the details.
- (c) Furthermore, for a feasible instance of FSC with harmonic capacities we present a polynomial algorithm as well as a strongly polynomial algorithm to compute the smallest feasible solution to FSC in time $\mathcal{O}(\min\{n^2 \log(a_n), n^3\}) \leq \mathcal{O}(n^3)$. See Section 3.2 for the details.
- (d) Our algorithm gives a strongly polynomial replacement for the heuristic component (which may fail to compute a solution) in the algorithm of Nguyen et al. [25]. However, we present an algorithm to solve the problem in linear time. See Section 4 for the details.

2 Notation and General Properties

For the sake of readability we write $X^{[\alpha]} = (X \bmod \alpha)$ for numbers X as well as $X^{[\alpha]} = \{z \bmod \alpha \mid z \in X\}$ for sets X (of numbers) to denote the modular projection of some number or interval, respectively. Extending the usual notation we also write $X \equiv Y \pmod{\alpha}$ if $X^{[\alpha]} = Y^{[\alpha]}$ for sets X, Y . Notice that on the one hand $(X \cup Y)^{[\alpha]} = X^{[\alpha]} \cup Y^{[\alpha]}$ but on the other hand be aware that $(X \cap Y)^{[\alpha]} \neq X^{[\alpha]} \cap Y^{[\alpha]}$ in general (cf. Lemma 9). Figure 1 depicts the structure of $v^{[\alpha]}$ if $v = [l_v, u_v]$ is an interval in \mathbb{Z} .

The empty set is denoted by \emptyset . Also we use the well-tried notation $t+X = \{t+z \mid z \in X\}$ to express the *translation* of a set of numbers X by some number t . For a set of sets \mathcal{S} we write $\bigcup \mathcal{S}$ to denote the union $\bigcup_{S \in \mathcal{S}} S$. Furthermore, we identify constraints by their indices. So for $i \leq n$ we say that “ $b_i \leq s + a_i x_i \leq B_i$ ” is constraint i .

Identity of BMS and FSC

It is important to notice that BMS allows zero capacities while FSC cannot allow zero divisors since $(\text{mod } 0)$ is undefined. However, consider a constraint i of BMS with $a_i \neq 0$. Let $b_i \leq s + a_i x_i \leq B_i$ be satisfied and set $r_i = s + a_i x_i$. Then $r_i^{[a_i]} = s^{[a_i]}$ and $r_i \in [b_i, B_i] = R_i$. Vice-versa let $r_i \in R_i$ s.t. $r_i \equiv s \pmod{a_i}$. Then there is an $x_i \in \mathbb{Z}$ s.t. $s + a_i x_i = r_i \in R_i = [b_i, B_i]$.

A constraint i that holds $a_i = 0$ simply demands that $s \in R_i$. Hence, if $a_i = a_j = 0$ for two constraints $i \neq j$ they can be replaced by one new constraint k defined by $R_k = R_i \cap R_j$. Therefore, one may assume that there is at most one constraint i with a zero capacity a_i . However, as all our results can be lifted back to the general case with low effort we will assume in terms of BMS that all capacities are non-zero and for FSC we make the equivalent assumption that $S = \mathbb{Z}_{\geq 0}$.

With our notation we may easily express the feasibility of a value s for a single constraint i as follows.

► **Observation 2.** *A value s satisfies constraint i if and only if $s^{[a_i]} \in R_i^{[a_i]}$.*

Proof. It holds that $\exists r_i \in R_i : r_i \equiv s \pmod{a_i}$ iff $\exists r_i \in R_i : r_i^{[a_i]} = s^{[a_i]}$ iff $s^{[a_i]} \in R_i^{[a_i]}$. ◀

By simply swapping the signs of the x_i we may assume that $a_i \geq 0$ for all i . We may also assume that the intervals are small in the sense that $B_i - b_i + 1 < a_i$ holds for all i . Assume that $B_i - b_i + 1 \geq a_i$ for an i and let $s \geq 0$ be an arbitrary integer. Then $b_i \leq B_i - a_i + 1$ and constraint i may always be solved by setting $x_i = \lceil (b_i - s)/a_i \rceil$ which satisfies

$$b_i \leq s + a_i \underbrace{\lceil \frac{b_i - s}{a_i} \rceil}_{x_i} \leq s + a_i \lceil \frac{B_i - a_i + 1 - s}{a_i} \rceil = s + a_i \lfloor \frac{B_i - s}{a_i} \rfloor \leq B_i.$$

Hence, constraint i is redundant and may be omitted. As a direct consequence there can be at most one feasible value for each x_i for a given guess s . In fact, we can decide the feasibility of a guess s in time $\mathcal{O}(n)$ as for all constraints i and values x_i it holds $b_i \leq s + a_i x_i \leq B_i$ if and only if $\lceil (b_i - s)/a_i \rceil = x_i = \lfloor (B_i - s)/a_i \rfloor$. So a guess s is feasible if and only if $\lceil (b_i - s)/a_i \rceil = \lfloor (B_i - s)/a_i \rfloor$ holds for all constraints i . Another consequence is that BMS is a generalization of MIXING SET as one can always add trivial upper bounds. By s_{\min} we denote the smallest feasible solution s that satisfies all constraints.

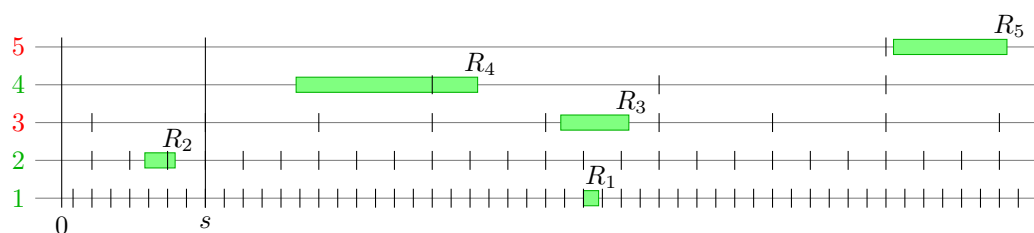
► **Observation 3.** *For feasible instances it holds that $s_{\min} < \text{lcm}(a_1, \dots, a_n)$.*

Proof. Let $\varphi = \text{lcm}(a_1, \dots, a_n)$. Remark that φ/a_i is integral for all i . Assume that (s, x) is a solution with $s = s_{\min} \geq \varphi$. Let $t = s - \varphi$ and $y_i = x_i + \varphi/a_i$ f.a. i . Then $0 \leq t < s_{\min}$ and $t + a_i y_i = s + a_i x_i$ f.a. i . So (t, y) is a solution that contradicts the optimality. ◀

3 Harmonic Divisors

Here we consider harmonic divisors in the sense that a_{i+1}/a_i is an integer for all $i < n$.

As we investigate some kind of a generalization of the setting of the Chinese Remainder Theorem, it is natural to ask for a CRT for *harmonic* (instead of the usually coprime) divisors and of course the (generalized) CRT answers this question; in this case we have $\text{gcd}(a_i, a_n) = a_i$ and so Theorem 1 reveals that if the system of n simultaneous congruences $s \equiv r_i \pmod{a_i}$ admits a solution then $r_i \equiv r_n \pmod{a_n}$ which says that if there is any solution then the set of all solutions is $a_n \mathbb{Z} + r_n^{[a_n]}$. However, it turns out that the investigation of FSC is a lot more complicated.



■ **Figure 2** $36a_1 = 18a_2 = 6a_3 = 3a_4 = a_5$. The guess s is not feasible for constr. 3 and 5.

In this section we present an algorithm to decide the feasibility of an instance of FSC. Also we show how optimal solutions can be computed in (strongly) polynomial time. Both of these results are based on the fine-grained interconnection between modular arithmetic on sets and the harmonic property. For some intuition Figure 2 gives a perspective on s as an anchor for 1-dimensional lattices with basis a_i which have to “hit” the intervals R_i . For example, in the figure it holds that $s + a_2 \cdot (-1) = s - a_2 \in R_2$, so the 1-dimensional lattice $(s + a_2z)_{z \in \mathbb{Z}}$ hits interval R_2 . Therefore, the choice of s satisfies constraint 2.

3.1 Deciding feasibility

The idea for our first algorithm will be to decide the feasibility problem by iteratively computing modular projections from constraint $i = n$ down to $i = 1$. In the following we will say that an interval $w \subseteq \mathbb{Z}$ represents a set $M \subseteq \mathbb{Z}$ (modulo α) if $w^{[\alpha]} = M^{[\alpha]}$. Also a set of intervals \mathcal{R} represents a set $M \subseteq \mathbb{Z}$ (modulo α) if $M^{[\alpha]} = \bigcup_{w \in \mathcal{R}} w^{[\alpha]}$. Given an integer $\alpha \geq 1$ and two intervals v, w we need to study the structure of the intersection $v^{[\alpha]} \cap w^{[\alpha]} \subseteq [0, \alpha)$. To express it let $v = [\ell_v, u_v], w = [\ell_w, u_w]$ and we define the basic intervals

$$\varphi_\alpha(v, w) = [\ell_v^{[\alpha]}, u_w^{[\alpha]}] \quad \text{and} \quad \psi_\alpha(v, w) = [\max\{\ell_v^{[\alpha]}, \ell_w^{[\alpha]}\}, \alpha + \min\{u_v^{[\alpha]}, u_w^{[\alpha]}\}]$$

for all intervals v, w . The former may be thought as the cases where $v^{[\alpha]}$ and $w^{[\alpha]}$ are two overlapping intervals while the intuition for the latter are situations where $v^{[\alpha]}$ and $w^{[\alpha]}$ both consist of two intervals which are in pairs overlapping. Remark that $\psi_\alpha(w, v) = \psi_\alpha(v, w)$ is always true.

► **Lemma 4.** *Given an integer $\alpha \geq 1$ and two intervals $v, w \subseteq \mathbb{Z}$ it holds that*

$$v^{[\alpha]} \cap w^{[\alpha]} \in \{ \emptyset, v^{[\alpha]}, w^{[\alpha]}, \psi_\alpha(v, w)^{[\alpha]}, \varphi_\alpha(v, w), \varphi_\alpha(w, v), \\ \varphi_\alpha(v, w) \dot{\cup} \varphi_\alpha(w, v), \varphi_\alpha(v, w) \dot{\cup} \psi_\alpha(v, w)^{[\alpha]}, \varphi_\alpha(w, v) \dot{\cup} \psi_\alpha(v, w)^{[\alpha]} \}.$$

The important intuition is that such a “modulo α intersection” can always be represented by at most two intervals. Remark that the sets in the second row are the only ones which are represented by $2 > 1$ intervals. Due to space reasons for the case distinction to prove Lemma 4 we refer to Appendix B and especially to Figure 6.

While Lemma 4 gives structure to intersections of two modular projections of intervals, the next lemma reveals how many intervals will be required to represent a *one-to-many* intersection. We will use this bound in every step of our algorithm. We want to add that both of these lemmas and even Lemma 6 do not depend on the harmonic property by themselves. However, they turn out to be especially useful in this setting.

► **Lemma 5.** *Let $\alpha \geq 1$, let v be an interval and let Q be a set of $k \geq 1$ intervals. Then there is a set R of at most $k+1$ intervals s.t. $v^{[\alpha]} \cap (\bigcup Q)^{[\alpha]} = (\bigcup R)^{[\alpha]}$.*

39:6 Fuzzy Simultaneous Congruences

Proof. We simply obtain that

$$v^{[\alpha]} \cap \left(\bigcup Q \right)^{[\alpha]} = \bigcup_{w \in Q} (v^{[\alpha]} \cap w^{[\alpha]}) = \bigcup_{\substack{w \in Q \\ w^{[\alpha]} \subseteq v^{[\alpha]}}} w^{[\alpha]} \cup \bigcup_{w \in D} (v^{[\alpha]} \cap w^{[\alpha]})$$

where $D = \{ w \in Q \mid w^{[\alpha]} \not\subseteq v^{[\alpha]}, w^{[\alpha]} \cap v^{[\alpha]} \neq \emptyset \}$ denotes the subset of intervals that cause the interesting intersections with $v^{[\alpha]}$ (cf. Lemma 4). Obviously, all other intersections can be represented by at most one interval each. So we study the intersections with D . In fact, everything gets simple if there are $w_1, w_2 \in D$ such that $v^{[\alpha]} \cap w_1^{[\alpha]} = \varphi_\alpha(v, w_1) \dot{\cup} \psi_\alpha(v, w_1)^{[\alpha]}$ and $v^{[\alpha]} \cap w_2^{[\alpha]} = \varphi_\alpha(w_2, v) \dot{\cup} \psi_\alpha(v, w_2)^{[\alpha]}$. By simply adapting the inequalities of the first case distinction in the proof of Lemma 4 we find

$$\begin{aligned} & (v^{[\alpha]} \cap w_1^{[\alpha]}) \cup (v^{[\alpha]} \cap w_2^{[\alpha]}) \\ &= ([0, u_v^{[\alpha]}] \dot{\cup} [\ell_v^{[\alpha]}, u_{w_1}^{[\alpha]}] \dot{\cup} [\ell_{w_1}^{[\alpha]}, \alpha]) \cup ([0, u_{w_2}^{[\alpha]}] \dot{\cup} [\ell_{w_2}^{[\alpha]}, u_v^{[\alpha]}] \dot{\cup} [\ell_v^{[\alpha]}, \alpha]) \\ &= [0, u_v^{[\alpha]}] \dot{\cup} [\ell_v^{[\alpha]}, \alpha] = v^{[\alpha]} \end{aligned}$$

which implies that $v^{[\alpha]} \cap \left(\bigcup Q \right)^{[\alpha]} = v^{[\alpha]}$ can be represented by only one interval, namely v . Therefore, in order to get an upper bound we assume that these two types of intersections do not come together. In more detail, we may assume by symmetry that $D = D_1 \dot{\cup} D_2$ where

$$\begin{aligned} D_1 &= \{ w \in D \mid v^{[\alpha]} \cap w^{[\alpha]} = \varphi_\alpha(v, w) \dot{\cup} \varphi_\alpha(w, v) \} \text{ and} \\ D_2 &= \{ w \in D \mid v^{[\alpha]} \cap w^{[\alpha]} = \varphi_\alpha(v, w) \dot{\cup} \psi_\alpha(v, w)^{[\alpha]} \}. \end{aligned}$$

It turns out that

$$\begin{aligned} \bigcup_{w \in D_1} (v^{[\alpha]} \cap w^{[\alpha]}) &= \bigcup_{w \in D_1} ([\ell_v^{[\alpha]}, u_w^{[\alpha]}] \dot{\cup} [\ell_w^{[\alpha]}, u_v^{[\alpha]}]) \\ &= [\ell_v^{[\alpha]}, \max_{w \in D_1} u_w^{[\alpha]}] \cup [\min_{w \in D_1} \ell_w^{[\alpha]}, u_v^{[\alpha]}] \text{ and} \\ \bigcup_{w \in D_2} (v^{[\alpha]} \cap w^{[\alpha]}) &= \bigcup_{w \in D_2} ([\ell_v^{[\alpha]}, u_w^{[\alpha]}] \dot{\cup} [\ell_w^{[\alpha]}, \alpha + u_v^{[\alpha]}]^{[\alpha]}) \\ &= [\ell_v^{[\alpha]}, \max_{w \in D_2} u_w^{[\alpha]}] \cup [\min_{w \in D_2} \ell_w^{[\alpha]}, \alpha + u_v^{[\alpha]}]^{[\alpha]} \end{aligned}$$

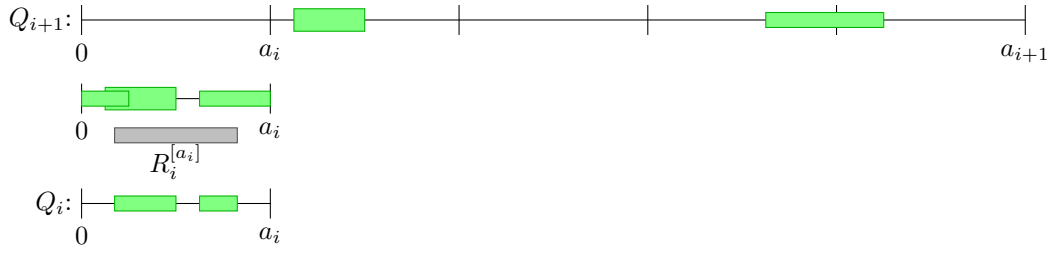
which finally joins up to

$$\bigcup_{w \in D} (v^{[\alpha]} \cap w^{[\alpha]}) = [\ell_v^{[\alpha]}, \max_{w \in D} u_w^{[\alpha]}] \cup [\min_{w \in D} \ell_w^{[\alpha]}, \alpha + u_v^{[\alpha]}]^{[\alpha]}.$$

Hence, all intersections with intervals in D may be represented by at most two intervals in total while each other intersection can be represented by at most one interval. Thus, if $|D| = 0$ then the whole intersection can be represented by at most k intervals. If $|D| \geq 1$ then there are at most $2 + |Q| - |D| \leq 2 + k - 1 = k + 1$ intervals required. \blacktriangleleft

Let S_i denote the set of all solutions $s \in \mathbb{Z}_{\geq 0}$ that are feasible for each of the constraints $i, i + 1, \dots, n$. We set $S_{n+1} = \mathbb{Z}_{\geq 0}$ to denote the feasible solutions to an empty set of constraints. The correctness of Algorithm 1 is implied by the following fundamental lemma. See Figure 3 for an example of a step inside the algorithm.

► Lemma 6. *It holds true that $S_i^{[a_i]} = R_i^{[a_i]} \cap S_{i+1}^{[a_i]}$ for all $i = 1, \dots, n$.*



■ **Figure 3** A step from $i+1$ to i ; modular projection to $[0, a_i)$ and intersection with $R_i^{[a_i]}$.

■ **Algorithm 1** Feasibility test for FSC.

procedure FEASIBLE($I = (a_1, \dots, a_n, R_1, \dots, R_n)$)

$Q_n \leftarrow \{R_n\}$

for $i = n - 1, \dots, 1$ **do**

 Compute set Q_i s. t. $(\bigcup Q_i)^{[a_i]} = R_i^{[a_i]} \cap (\bigcup Q_{i+1})^{[a_i]}$ and $|Q_i| \leq \mathcal{O}(n - i)$

if $\bigcup Q_1 = \emptyset$ **then**

return “infeasible”

else

return “feasible”

Proof. Let $r \in S_i^{[a_i]}$. So there is a solution $s \in S_i$ such that $r = s^{[a_i]} \in R_i^{[a_i]}$. It holds that $S_i \subseteq S_{i+1}$ which implies $s \in S_{i+1}$ and thus $r = s^{[a_i]} \in S_{i+1}^{[a_i]}$.

Vice-versa let $r \in R_i^{[a_i]} \cap S_{i+1}^{[a_i]}$. So there is a solution $s \in S_{i+1}$ with $s^{[a_i]} = r$. From $r \in R_i^{[a_i]}$ we get $s^{[a_i]} \in R_i^{[a_i]}$. Hence, $s \in S_i$ and $r = s^{[a_i]} \in S_i^{[a_i]}$. ◀

▶ **Theorem 7.** *Algorithm 1 decides the feasibility of an instance in time $\mathcal{O}(n^2)$.*

Proof. We show that $\bigcup Q_i \equiv S_i \pmod{a_i}$ for all $i = n, \dots, 1$. This will prove the algorithm correct since then $\bigcup Q_1 \equiv S_1 \pmod{a_1}$ and that means $\bigcup Q_1$ is empty if and only if S_1 is empty. Obviously it holds that $\bigcup Q_n \equiv S_n \pmod{a_n}$ since $\bigcup Q_n = R_n$. Now suppose that $\bigcup Q_{i+1} \equiv S_{i+1} \pmod{a_{i+1}}$ for some $i \geq 1$. We have that $(\bigcup Q_i)^{[a_i]} = R_i^{[a_i]} \cap (\bigcup Q_{i+1})^{[a_i]}$ where the harmonic property implies $(\bigcup Q_{i+1})^{[a_i]} = ((\bigcup Q_{i+1})^{[a_{i+1}]})^{[a_i]} = (S_{i+1}^{[a_{i+1}]})^{[a_i]} = S_{i+1}^{[a_i]}$. Together with Lemma 6 this yields $(\bigcup Q_i)^{[a_i]} = R_i^{[a_i]} \cap S_{i+1}^{[a_i]} = S_i^{[a_i]}$ and that proves the algorithm correct. Using Lemmas 4–6 each set Q_i can be computed in time $\mathcal{O}(n)$ and this yields a total running time of $\mathcal{O}(n^2)$. ◀

3.2 Optimal solutions

Unfortunately, Algorithm 1 neither calculates a solution nor it directly implies one. Here we present an algorithm to compute the smallest feasible solution s_{\min} to FSC. However, by searching in the opposite direction the same technique also applies to the computation of the largest feasible solution $s_{\max} < a_n$. We start with a simple binary search approach.

▶ **Corollary 8.** *For feasible instances s_{\min} can be computed in time $\mathcal{O}(n^2 \log(a_n))$.*

This can be achieved by introducing an additional constraint measuring the value of s as follows. Let β be a positive integer. We extend the problem instance by a new constraint with number $n + 1$ defined by $a_{n+1} = 2 \cdot a_n$, $b_{n+1} = 0$, and $B_{n+1} = \beta$. Remark that this

β -instance admits the same set of solutions as the original instance as long as β is large enough, e.g. $\beta = a_n$ (cf. Observation 3). Consider a feasible solution to the β -instance where $\beta \leq a_n$. It holds that

$$2a_n x_{n+1} = a_{n+1} x_{n+1} \leq s + a_{n+1} x_{n+1} \leq B_{n+1} = \beta \leq a_n$$

which implies $x_{n+1} \leq \lfloor \frac{1}{2} \rfloor = 0$. However, if $x_{n+1} < 0$ then $s \geq a_{n+1} \cdot |x_{n+1}|$ and therefore the solution $s' = s + a_{n+1} x_{n+1}$ with $x'_{n+1} = 0$ and $x'_i = x_i - (a_{n+1}/a_i)x_{n+1}$ for all $i = 1, \dots, n$ is better than s and $x'_{n+1} = 0$.

Thus we may assume generally that $x_{n+1} = 0$ which allows us to measure the value of s using the upper bound β . We use β to do a binary search in the interval $[0, a_n]$ using Algorithm 1 to check the β -instance for feasibility. The smallest possible value for β then states the optimum value and that proves Corollary 8. However, with additional ideas we are able to achieve strongly polynomial time. We want to give some helpful intuition first.

Clearly, after revealing the intervals in Q_1 with Algorithm 1 a straightforward idea is to try tracing them back to a small solution for s , but routing through the modulus operations appears to become a non-polynomial bottleneck.

However, the following idea is a first step to end up with a constraint aggregation approach. Given the projections $A^{[ab]}$ and $B^{[a]}$ of two sets $A, B \subseteq \mathbb{Z}$ one can compute the intersection $A^{[a]} \cap B^{[a]}$ in at least two ways; primitively we compute $(A^{[ab]})^{[a]} = A^{[a]}$ and then intersect it with $B^{[a]}$, but also we can intersect $A^{[ab]}$ with b translated copies $B^{[a]}, a + B^{[a]}, \dots, (b-1)a + B^{[a]}$ of $B^{[a]}$ before computing the $[a]$ -projection. In fact, the following lemma seems to be a characteristic property of modular arithmetic on sets.

► **Lemma 9.** For all numbers $a, b \in \mathbb{Z}_{\geq 1}$ and sets $A, B \subseteq \mathbb{Z}$ it holds

$$A^{[a]} \cap B^{[a]} = \left(A^{[ab]} \cap \bigcup_{i=0}^{b-1} (ia + B^{[a]}) \right)^{[a]}.$$

Proof. Let x be a number. Then it holds

$$\begin{aligned} x \in \left(A^{[ab]} \cap \bigcup_{i=0}^{b-1} (ia + B^{[a]}) \right)^{[a]} &\Leftrightarrow \exists y \in A^{[ab]} : y \in \bigcup_{i=0}^{b-1} (ia + B^{[a]}) \wedge x = y^{[a]} \\ &\Leftrightarrow \exists y \in A^{[ab]} : y^{[a]} \in B^{[a]} \wedge x = y^{[a]} \\ &\Leftrightarrow x \in A^{[a]} \cap B^{[a]} \end{aligned}$$

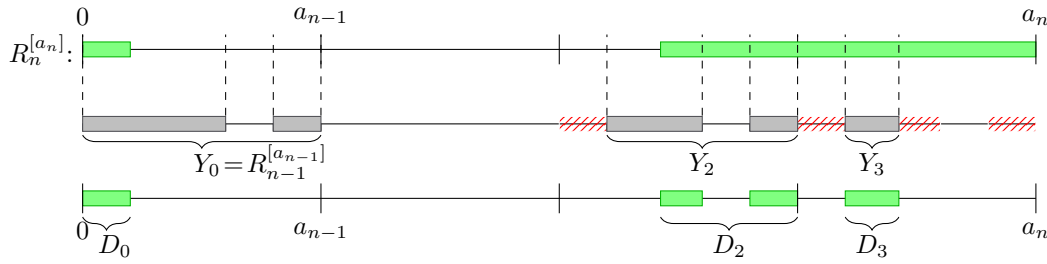
where the last equivalence follows from $(A^{[ab]})^{[a]} = A^{[a]}$. ◀

Since the right side can be written as the modular projection of a *union of intersections* we can find a sensible strengthening; in fact, for arbitrary sets X, M_0, \dots, M_{m-1} it holds that

$$\bigcup_{i=0}^{m-1} (X \cap M_i) = \bigcup_{i=0}^{m-1} (X \cap (M_i \setminus \bigcup_{j=0}^{i-1} (X \cap M_j))).$$

While the left-hand side may not, the right-hand side is always a *disjoint* union. Taking into account the modular projections this leads to the following corollary.

► **Corollary 10.** For all numbers $a, b \in \mathbb{Z}_{\geq 1}$ and sets $A, B \subseteq \mathbb{Z}$ it holds $A^{[a]} \cap B^{[a]} = (\bigcup_{i=0}^{b-1} D_i)^{[a]}$ where $D_i = A^{[ab]} \cap Y_i$ and $Y_i = ia + (B^{[a]} \setminus \bigcup_{j=0}^{i-1} D_j^{[a]})$ for all $i = 0, \dots, b-1$.



■ **Figure 4** An example of four required intervals to represent $R_{n-1}^{[a_{n-1}]} \cap R_n^{[a_n]}$ in Lemma 13.

We will use Corollary 10 to aggregate constraints in order to reduce the problem size. The following observation gives a first bound for the smallest feasible solution s_{\min} .

► **Observation 11.** *For feasible instances it holds that $s_{\min} \in R_n^{[a_n]}$.*

This is true since in the harmonic case $s_{\min} < \text{lcm}(a_1, \dots, a_n) = a_n$ due to Observation 3 which then implies that $s_{\min} = s_{\min}^{[a_n]} \in R_n^{[a_n]}$ using Observation 2. Motivated by Observation 11 the idea is to search for s_{\min} in the modular projection $R_n^{[a_n]}$ by aggregating the penultimate constraint $n - 1$ into the last constraint n . In fact, the number of required intervals to represent both constraints can be bounded by a constant. A fine-grained construction then enforces the algorithm to efficiently iterate the feasibility test on aggregated instances to find the optimum value.

► **Theorem 12.** *For feasible instances s_{\min} can be computed in time $\mathcal{O}(n^3)$.*

Remark that the set of feasible solutions for the last two constraints is $S_{n-1} = R_{n-1}^{[a_{n-1}]} \cap (R_n^{[a_n]})^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]} \cap R_n^{[a_{n-1}]}$. Therefore, the next lemma states the crucial argument of the algorithm.

► **Lemma 13.** *The intersection $R_{n-1}^{[a_{n-1}]} \cap R_n^{[a_{n-1}]}$ can always be represented by the disjoint union $U \subseteq R_n^{[a_n]}$ of only constant many intervals in $R_n^{[a_n]}$ such that*

(a) $U^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]} \cap R_n^{[a_{n-1}]}$ and

(b) $u \equiv r \pmod{a_{n-1}}$ implies $u \leq r$ for all $u \in U$, $r \in R_n^{[a_n]}$.

Here the former property states that indeed the intervals in U are a proper representation for the last two constraints. The important property is the latter; in fact, it ensures that U is the best possible representation in the sense that U consists of the *smallest* intervals possible (see Figure 4).

Proof of Lemma 13. (a). By defining $D_i = Y_i \cap R_n^{[a_n]}$ and

$$Y_i = ia_{n-1} + (R_{n-1}^{[a_{n-1}]} \setminus \bigcup_{j=0}^{i-1} D_j^{[a_{n-1}]})$$

for all $i \in \{0, \dots, a_n/a_{n-1} - 1\}$ Corollary 10 proves the claim (cf. Figure 4). (b) follows by construction.

It remains to show that $\bigcup_i D_i$ is the union of only constant many disjoint intervals. Apparently, the intervals are disjoint by construction.

We claim that there are at most three non-empty sets D_i . Assume there are at least four non-empty translates D_i , namely D_i, D_j, D_k, D_ℓ . Then, since R_n is an interval it holds for at least two $p, q \in \{i, j, k, \ell\}$ that the *full interval translates* $F_p = [pa_{n-1}, (p+1)a_{n-1})$ and $F_q = [qa_{n-1}, (q+1)a_{n-1})$ are subsets of $R_n^{[a_n]}$. For p (and also for q) we get

$$D_p^{[a_{n-1}]} = \underbrace{(Y_p \cap R_n^{[a_n]})^{[a_{n-1}]}}_{\subseteq F_p} = Y_p^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]} \setminus \bigcup_{j=0}^{p-1} D_j^{[a_{n-1}]}$$

which implies with $\bigcup_{j=0}^{p-1} D_j^{[a_{n-1}]} \subseteq R_{n-1}^{[a_{n-1}]}$ that

$$\bigcup_{j=0}^p D_j^{[a_{n-1}]} = D_p^{[a_{n-1}]} \cup \bigcup_{j=0}^{p-1} D_j^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]}.$$

Then it follows $\bigcup_{j=0}^p D_j^{[a_{n-1}]} = R_{n-1}^{[a_{n-1}]} = \bigcup_{j=0}^q D_j^{[a_{n-1}]}$. W.l.o.g. let $p < q$. Then $D_q = Y_q \cap R_n^{[a_n]}$ is empty since

$$Y_q = qa_{n-1} + \left(R_{n-1}^{[a_{n-1}]} \setminus \bigcup_{j=0}^{q-1} D_j^{[a_{n-1}]} \right) \subseteq qa_{n-1} + \left(R_{n-1}^{[a_{n-1}]} \setminus R_{n-1}^{[a_{n-1}]} \right)$$

is empty and we have a contradiction.

Using the same case distinctions as in the proof of Lemma 4 one can show that each set D_i consist of at most two intervals. Therefore, all the non-empty sets D_i consist of at most $3 \cdot 2 = 6$ intervals in total. In fact, one can improve this bound to a total number of at most 4 intervals (see Figure 4) by a more sophisticated case distinction. ◀

This admits an algorithm using an aggregation argument as follows. For constraints n and $n - 1$ we use Lemma 13 to compute disjoint intervals $E_1, \dots, E_k \subseteq R_n^{[a_n]}$ (representing the constraints n and $n - 1$) where $k \leq C$ for a small constant C . If $k \geq 1$ then use Algorithm 1 to check the feasibility of the instances $\mathcal{I}_1, \dots, \mathcal{I}_k$ defined by

$$(\mathcal{I}_j) \quad \min \{ s \mid s^{[a_i]} \in R_i^{[a_i]} \forall i = 1, \dots, n-2, s^{[a_n]} \in E_j^{[a_n]}, s \in \mathbb{Z}_{\geq 0} \}.$$

If none of the instances $\mathcal{I}_1, \dots, \mathcal{I}_k$ admits a solution then the original instance can not be feasible. Assume that there is at least one feasible instance. Now, since E_1, \dots, E_k are disjoint exactly one of them contains the optimum value for s . W.l.o.g. assume that $E_1 < \dots < E_k$. Then there is a smallest index j such that \mathcal{I}_j is feasible and we solve \mathcal{I}_j recursively to find the optimum value. Together this yields an algorithm running in time $n \cdot C \cdot \mathcal{O}(n^2) = \mathcal{O}(n^3)$.

4 Uniprocessor Real-Time Scheduling

In real-time systems an important question is to ask for the *worst-case response time* of a task system. While the complexity is pseudo-polynomial in general [28], Nguyen et al. proposed a new algorithm [25] to compute it in polynomial time for preemptive sporadic tasks τ_1, \dots, τ_n with harmonic periods $T_i \geq 0$ and job processing times $C_i \geq 0$ running on a uniprocessor platform. The worst-case response time is the first point in time where $t = C_n + \sum_{i=1}^{n-1} C_i \cdot \lceil t/T_i \rceil$. Be aware that they assume the harmonic property in the opposite direction, i.e. $T_i/T_{i+1} \in \mathbb{Z}$. Their algorithm even allows the task execution to be delayed by some release jitter J_i . However, their algorithm depends on a heuristic component which may fail to compute correct solutions [25, Section 5.5, 6]. In fact, the fundamental computation problem can be expressed as a BMS instance which immediately implies a robust approach in time $\mathcal{O}(n^3)$ with our algorithm. Nevertheless, it can be solved even more efficiently in time $\mathcal{O}(n)$ which we describe here. The overall result will be the following theorem.

► **Theorem 14.** *The worst-case response time of a harmonic task real-time system with constrained release jitter can be computed in polynomial time.*

We adapt the notation of Nguyen et al. and extend it to our needs. The jobs of task τ_i have the processing time C_i and we define $c_i = \sum_{t=i+1}^{n-1} C_t$ to accumulate the last of them. The utilization of task τ_i is denoted by $U_i = C_i/T_i$ and it holds that $\sum_{t=1}^{n-1} U_t < 1$. In [25, Section 5.4.1] Nguyen et al. describe that also $x_1 = 1$ may be assumed. The system to solve (eq. (55), (56)) is described in [25, Section 5.5]:

$$\begin{aligned} \min \{ x_n \mid & J_i + T_i x_i \leq J_n + T_n x_n, \\ & J_n + T_n x_n - c_i \leq J_i + T_i x_i \quad \forall i \leq n-1 \} \end{aligned} \quad (2)$$

which can be formulated as the following BMS instance:

$$\min \left\{ x_n \mid \left\lceil \frac{J_i - J_n}{T_n} \right\rceil \leq x_n - \frac{T_i}{T_n} x_i \leq \left\lfloor \frac{J_i - J_n + c_i}{T_n} \right\rfloor \quad \forall i \leq n-1 \right\} \quad (3)$$

► **Lemma 15.** *If $i < j \leq n$ and $(c_i + c_j)/T_j < 1$ then in terms of variable x_i there is at most one feasible value for variable x_j .*

Proof. If $j < n$ then by combining the constraints for i and j in (2) we find

$$\begin{aligned} T_i x_i + J_i - J_n &\leq T_j x_j + J_j - J_n + c_j \quad \text{and} \\ T_j x_j + J_j - J_n &\leq T_i x_i + J_i - J_n + c_i \end{aligned}$$

which with the harmonic property and the integrality of x_j yields

$$\frac{T_i}{T_j} x_i + \left\lceil \frac{J_i - J_j - c_j}{T_j} \right\rceil \leq x_j \leq \frac{T_i}{T_j} x_i + \left\lfloor \frac{J_i - J_j + c_i}{T_j} \right\rfloor. \quad (4)$$

However, if $j = n$ then $c_j = \sum_{t=n+1}^{n-1} C_t = 0$ and thus (4) follows from (2) too (cf. (3)). Now by simply dropping the roundings we obtain in both cases that

$$\frac{T_i}{T_j} x_i + \left\lfloor \frac{J_i - J_j + c_i}{T_j} \right\rfloor - \left(\frac{T_i}{T_j} x_i + \left\lceil \frac{J_i - J_j - c_j}{T_j} \right\rceil \right) \leq \frac{c_i + c_j}{T_j} < 1$$

which proves the claim. ◀

According to (4) we define interval bounds $\ell_j^{(i)}(z)$ and $u_j^{(i)}(z)$ to denote the feasible values for variable x_j in terms of variable x_i where z states a value for variable x_i , i.e.

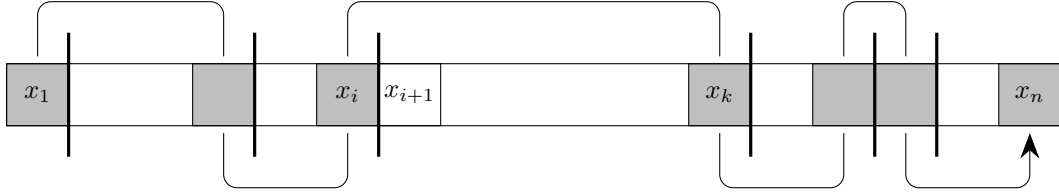
$$\ell_j^{(i)}(z) = \frac{T_i}{T_j} z + \left\lceil \frac{J_i - J_j - c_j}{T_j} \right\rceil \quad \text{and} \quad u_j^{(i)}(z) = \frac{T_i}{T_j} z + \left\lfloor \frac{J_i - J_j + c_i}{T_j} \right\rfloor.$$

Thus, (4) is equivalent to $x_j \in [\ell_j^{(i)}(x_i), u_j^{(i)}(x_i)]$ and if $(c_i + c_j)/T_j < 1$ then it either holds that $\ell_j^{(i)}(x_i) = x_j = u_j^{(i)}(x_i)$ or there is no solution at all.

Fortunately, there is always a sequence of variables such that the value of every next variable can be determined by the value of the current variable. The following lemma is crucial.

► **Lemma 16.** *If $i < n$ and $k = \max \{ t \leq n \mid T_{i+1} = T_t \}$ then there is at most one feasible value for variable x_k .*

39:12 Fuzzy Simultaneous Congruences



■ **Figure 5** The variable revealing flow with vertical lines between blocks of equal periods.

Proof. If $k < n - 1$ then it holds by the harmonic property and the maximality of k that $T_k \geq 2T_{k+1} \geq 2T_{k+2} \geq \dots \geq 2T_{n-1}$ and thus $T_t/T_k \leq 1/2$ for all $t = k + 1, \dots, n - 1$. Hence,

$$\frac{c_i + c_k}{T_k} = \sum_{t=i+1}^{n-1} U_t \frac{T_t}{T_k} + \sum_{t=k+1}^{n-1} U_t \frac{T_t}{T_k} = \sum_{t=i+1}^k U_t \underbrace{\frac{T_t}{T_k}}_{=1} + 2 \sum_{t=k+1}^{n-1} U_t \underbrace{\frac{T_t}{T_k}}_{\leq 1/2} \leq \sum_{t=i+1}^{n-1} U_t < 1.$$

If otherwise $k \geq n - 1$ then $c_k = 0$ and hence

$$\frac{c_i + c_k}{T_k} = \frac{c_i}{T_k} = \sum_{t=i+1}^{n-1} U_t \underbrace{\frac{T_t}{T_k}}_{=1} = \sum_{t=i+1}^{n-1} U_t < 1.$$

By Lemma 15 this proves the claim. ◀

This gives rise to the following algorithm. By iterating Lemma 16 and starting with $x_1 = 1$ we can reveal the last variable of each block of indices of equal periods (cf. Figure 5). Finally, this reveals the variable x_n and we only need to assure that the value of x_n admits feasible values for variables which are not revealed so far. Apparently we may restate the constraints of (2) as

$$\left\lfloor \frac{J_n - J_j - c_j + T_n x_n}{T_j} \right\rfloor \leq x_j \leq \left\lfloor \frac{J_n - J_j + T_n x_n}{T_j} \right\rfloor \quad \forall j = 1, \dots, n - 1.$$

Therefore, we can simply compare these bounds to assure the existence of a feasible value for each variable x_j . See Algorithm 2 for a formal description.

■ **Algorithm 2** Variable revealing flow.

```

procedure REVEAL
   $x_1 \leftarrow 1$ 
   $k \leftarrow 1$ 
  while  $k < n$  do
     $i \leftarrow k$ 
     $k \leftarrow \max \{ t \leq n \mid T_{i+1} = T_t \}$ 
    if  $\ell_k^{(i)}(x_i) \neq u_k^{(i)}(x_i)$  then
      return -1
    else
       $x_k \leftarrow \ell_k^{(i)}(x_i)$  ▷ Lemma 16
    for  $j = 1, \dots, n - 1$  do
      if  $\left\lfloor \frac{J_n - J_j - c_j + T_n x_n}{T_j} \right\rfloor > \left\lfloor \frac{J_n - J_j + T_n x_n}{T_j} \right\rfloor$  then ▷ no feasible solution for  $x_j$ 
      return -1
  return  $x_n$ 

```

► **Observation 17.** *By a more sophisticated investigation the number of index blocks of equal periods can be bounded by a constant and thus, the `while` loop reveals x_n in constant time. Therefore, the final feasibility test appears to be the only computational bottleneck.*

References

- 1 Manindra Agrawal and Somenath Biswas. Primality and identity testing via chinese remaindering. In *Proc. FOCS 1999*, pages 202–209, 1999.
- 2 Saoussen Anssi, Stefan Kuntz, Sébastien Gérard, and François Terrier. On the gap between schedulability tests and an automotive task model. *J. Syst. Archit.*, 59(6):341–350, 2013.
- 3 Reuven Bar-Yehuda and Dror Rawitz. Efficient algorithms for integer programs with two variables per constraint. *Algorithmica*, 29(4):595–609, 2001.
- 4 Sanjoy K. Baruah, Louis E. Rosier, and Rodney R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, 1990.
- 5 Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, Nicole Megow, and Andreas Wiese. Polynomial-time exact schedulability tests for harmonic real-time tasks. In *Proc. RTSS 2013*, pages 236–245, 2013.
- 6 Michele Conforti, Gerard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Springer, 2014.
- 7 Michele Conforti, Marco Di Summa, and Laurence A. Wolsey. The mixing set with flows. *SIAM J. Discrete Math.*, 21(2):396–407, 2007.
- 8 Michele Conforti, Marco Di Summa, and Laurence A. Wolsey. The mixing set with divisible capacities. In *Proc. IPCO 2008*, pages 435–449, 2008.
- 9 Michele Conforti and Giacomo Zambelli. The mixing set with divisible capacities: A simple approach. *Oper. Res. Lett.*, 37(6):379–383, 2009.
- 10 Jana Cslovjceksek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time, 2020 (Manuscript). [arXiv:2002.07745](https://arxiv.org/abs/2002.07745).
- 11 Friedrich Eisenbrand, Karthikeyan Kesavan, Raju S. Mattikalli, Martin Niemeier, Arnold W. Nordsieck, Martin Skutella, José Verschae, and Andreas Wiese. Solving an avionics real-time scheduling problem by advanced ip-methods. In Mark de Berg and Ulrich Meyer, editors, *Proc. ESA 2010*, volume 6346 of *Lecture Notes in Computer Science*, pages 11–22, 2010.
- 12 Friedrich Eisenbrand and Günter Rote. Fast 2-variable integer programming. In *Proc. IPCO 2001*, pages 78–89, 2001.
- 13 Friedrich Eisenbrand and Thomas Rothvoß. New hardness results for diophantine approximation. In *Proc. APPROX 2009*, pages 98–110, 2009.
- 14 Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese remaindering with errors. In *Proc. STOC 1999*, pages 225–234, 1999.
- 15 Oktay Günlük and Yves Pochet. Mixing mixed-integer inequalities. *Math. Program.*, 90(3):429–457, 2001.
- 16 Venkatesan Guruswami, Amit Sahai, and Madhu Sudan. "soft-decision" decoding of chinese remainder codes. In *Proc. FOCS 2000*, pages 159–168, 2000.
- 17 Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. n-fold integer programming in cubic time. *Math. Program.*, 137(1-2):325–341, 2013.
- 18 Raymond Hemmecke and Rüdiger Schultz. Decomposition of test sets in stochastic integer programming. *Math. Program.*, 94(2-3):323–341, 2003.
- 19 Klaus Jansen, Alexandra Lassota, and Lars Rohwedder. Near-linear time algorithm for n-fold ilps via color coding. In *Proc. ICALP 2019*, pages 75:1–75:13, 2019.
- 20 Kim-Manuel Klein. About the complexity of two-stage stochastic ips. In Daniel Bienstock and Giacomo Zambelli, editors, *Proc. IPCO 2020*, volume 12125 of *Lecture Notes in Computer Science*, pages 252–265, 2020.

- 21 Donald E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition*. Prentice Hall, 1981.
- 22 J. C. Lagarias. The computational complexity of simultaneous diophantine approximation problems. *SIAM J. Comput.*, 14(1):196–209, 1985.
- 23 Joseph Y.-T. Leung and Jennifer Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Perform. Evaluation*, 2(4):237–250, 1982.
- 24 Andrew J. Miller and Laurence A. Wolsey. Tight formulations for some simple mixed integer programs and convex objective integer programs. *Math. Program.*, 98(1-3):73–88, 2003.
- 25 Thi Huyen Chau Nguyen, Werner Grass, and Klaus Jansen. Exact polynomial time algorithm for the response time analysis of harmonic tasks with constrained release jitter, 2019 (Manuscript). [arXiv:1912.01161](https://arxiv.org/abs/1912.01161).
- 26 Yves Pochet and Laurence A. Wolsey. *Production Planning by Mixed Integer Programming (Springer Series in Operations Research and Financial Engineering)*. Springer, Berlin, Heidelberg, 2006.
- 27 Chi-Sheng Shih, Sathish Gopalakrishnan, Phanindra Ganti, Marco Caccamo, and Lui Sha. Template-based real-time dwell scheduling with energy constraint. In *Proc. RTAS 2003*, page 19, 2003.
- 28 Mikael Sjödin and Hans Hansson. Improved response-time analysis calculations. In *Proc. RTSS 1998*, pages 399–408, 1998.
- 29 Yang Xu, Anton Cervin, and Karl-Erik Årzén. Lqg-based scheduling and control co-design using harmonic task periods. Technical report, Department of Automatic Control, Lund Institute of Technology, Lund University, August 2016. URL: https://lup.lub.lu.se/search/ws/files/10751577/bare_conf.pdf.
- 30 Ming Zhao and Ismael R. de Farias Jr. The mixing-mir set with divisible capacities. *Math. Program.*, 115(1):73–103, 2008.

A Hardness of BMS

We reduce from DIRECTED DIOPHANTINE APPROXIMATION with *rounding down*. For any vector $v \in \mathbb{R}^n$ let $\lfloor v \rfloor$ denote the vector where each component is rounded down, i.e. $(\lfloor v \rfloor)_i = \lfloor v_i \rfloor$ for all $i \leq n$.

DIRECTED DIOPHANTINE APPROXIMATION with rounding down (DDA[↓])
 Given: $\alpha_1, \dots, \alpha_n \in \mathbb{Q}_+$, $N \in \mathbb{Z}_{\geq 1}$, $\varepsilon \in \mathbb{Q}$, $0 < \varepsilon < 1$
 Decide whether there is a $Q \in \{1, \dots, N\}$ such that $\|Q\alpha - \lfloor Q\alpha \rfloor\|_\infty \leq \varepsilon$.

Eisenbrand and Rothvoß proved that DDA[↓] is NP-hard [13]. In fact, every instance of DDA[↓] can be expressed as a BMS instance, which yields the following theorem.

► **Theorem 18.** BMS is NP-hard (even if $b_i = 0$ for all i with $a_i \neq 0$).

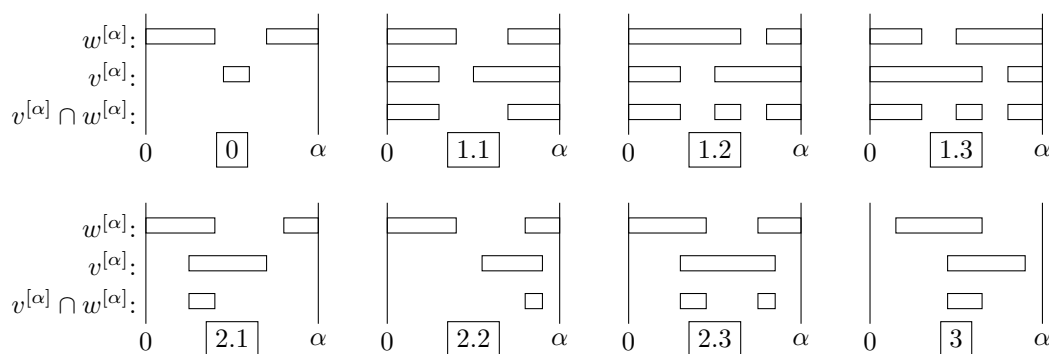
Proof. Write $\alpha_i = \beta_i/\gamma_i$ for integers $\beta_i \geq 0, \gamma_i \geq 1$ and set $\lambda = \prod_j \beta_j$. Then $\lambda/\alpha_i = (\lambda/\beta_i)\gamma_i \geq 0$ is integer. Let \mathcal{M} denote the following instance of BMS:

$$0 \leq Q' - (\lambda/\alpha_i) \cdot y_i \leq \lfloor (\lambda/\alpha_i) \cdot \varepsilon \rfloor \quad \forall i = 1, \dots, n \quad (5)$$

$$\lambda \leq Q' - 0 \cdot y_{n+1} \leq \lambda \cdot N \quad (6)$$

$$0 \leq Q' - \lambda \cdot y_{n+2} \leq 0 \quad (7)$$

$$Q', y_i \in \mathbb{Z} \quad \forall i = 1, \dots, n+2$$



■ **Figure 6** Examples for the cases of the case distinction in the proof of Lemma 4.

So let $Q \in \{1, \dots, N\}$ with $\|Q\alpha - \lfloor Q\alpha \rfloor\|_\infty \leq \varepsilon$ be given. We obtain readily that $Q' = \lambda Q$ and $y = (\lfloor Q\alpha_1 \rfloor, \dots, \lfloor Q\alpha_n \rfloor, 0, Q)$ defines a solution of \mathcal{M} since

$$0 \leq Q\alpha_i - \lfloor Q\alpha_i \rfloor \leq \varepsilon \quad \text{if and only if} \quad 0 \leq \underbrace{\lambda Q - (\lambda/\alpha_i) \cdot \lfloor Q\alpha_i \rfloor}_{\in \mathbb{Z}} \leq (\lambda/\alpha_i) \cdot \varepsilon.$$

Vice-versa let (Q', y) be a solution to \mathcal{M} . We see that (5) implies that

$$0 \leq Q' - (\lambda/\alpha_i) \cdot y_i \leq \lfloor (\lambda/\alpha_i) \cdot \varepsilon \rfloor \leq (\lambda/\alpha_i) \cdot \varepsilon$$

and by (7) we get $Q' = \lambda \cdot y_{n+2}$ which then implies $0 \leq y_{n+2}\alpha_i - y_i \leq \varepsilon < 1$ for all $i \leq n$. Now, since y_i is integer, there can be only one value for y_i , i.e. $y_i = \lfloor y_{n+2}\alpha_i \rfloor$. By $Q' = \lambda \cdot y_{n+2}$ and (6) we get $y_{n+2} \in \{1, \dots, N\}$ and by setting $Q = y_{n+2}$ this yields $\|Q\alpha - \lfloor Q\alpha \rfloor\|_\infty \leq \varepsilon$ and that proves the claim. ◀

B Omitted proofs

Proof of Lemma 4. We do a case distinction (see Figure 6) as follows. We only look at the non-trivial case, i.e. $v^{[\alpha]} \cap w^{[\alpha]} \notin \{\emptyset, v^{[\alpha]}, w^{[\alpha]}\}$, which especially implies $|v| < \alpha$ and $|w| < \alpha$.

We start with the case that neither $v^{[\alpha]}$ nor $w^{[\alpha]}$ is an interval, i.e. $u_v^{[\alpha]} < \ell_v^{[\alpha]}$ and $u_w^{[\alpha]} < \ell_w^{[\alpha]}$. Then it cannot be that $u_w^{[\alpha]} \geq \ell_v^{[\alpha]}$ and $u_v^{[\alpha]} \geq \ell_w^{[\alpha]}$ since that implies $\ell_v^{[\alpha]} \leq u_w^{[\alpha]} < \ell_w^{[\alpha]} \leq u_v^{[\alpha]}$. Hence, there are three cases as follows.

CASE 1.1. $u_w^{[\alpha]} < \ell_v^{[\alpha]}$ and $u_v^{[\alpha]} < \ell_w^{[\alpha]}$. Then the intersection equals

$$\begin{aligned} [0, \min\{u_v^{[\alpha]}, u_w^{[\alpha]}\}] \cup [\max\{\ell_v^{[\alpha]}, \ell_w^{[\alpha]}\}, \alpha] &= [\max\{\ell_v^{[\alpha]}, \ell_w^{[\alpha]}\}, \alpha + \min\{u_v^{[\alpha]}, u_w^{[\alpha]}\}]^{[\alpha]} \\ &= \psi_\alpha(v, w)^{[\alpha]}. \end{aligned}$$

CASE 1.2. $u_w^{[\alpha]} \geq \ell_v^{[\alpha]}$ and $u_v^{[\alpha]} < \ell_w^{[\alpha]}$. Then the intersection equals

$$[0, u_v^{[\alpha]}] \cup [\ell_v^{[\alpha]}, u_w^{[\alpha]}] \cup [\ell_w^{[\alpha]}, \alpha] = [\ell_v^{[\alpha]}, u_w^{[\alpha]}] \cup [\ell_w^{[\alpha]}, \alpha + u_v^{[\alpha]}]^{[\alpha]} = \varphi_\alpha(v, w) \cup \psi_\alpha(v, w)^{[\alpha]}.$$

CASE 1.3. $u_w^{[\alpha]} < \ell_v^{[\alpha]}$ and $u_v^{[\alpha]} \geq \ell_w^{[\alpha]}$. By symmetry we get $v^{[\alpha]} \cap w^{[\alpha]} = \varphi_\alpha(w, v) \cup \psi_\alpha(v, w)^{[\alpha]}$.

Now, w.l.o.g. assume that $v^{[\alpha]}$ is an interval, i.e. $\ell_v^{[\alpha]} \leq u_v^{[\alpha]}$, while $w^{[\alpha]}$ consists of two intervals, i.e. $u_w^{[\alpha]} < \ell_w^{[\alpha]}$. Then there are three cases as follows.

39:16 Fuzzy Simultaneous Congruences

CASE 2.1. $\ell_v^{[\alpha]} \leq u_w^{[\alpha]} < u_v^{[\alpha]} < \ell_w^{[\alpha]}$. Then the intersection equals $[\ell_v^{[\alpha]}, u_w^{[\alpha]}] = \varphi_\alpha(v, w)$.

CASE 2.2. $u_w^{[\alpha]} < \ell_v^{[\alpha]} < \ell_w^{[\alpha]} \leq u_v^{[\alpha]}$. Then the intersection equals $[\ell_w^{[\alpha]}, u_v^{[\alpha]}] = \varphi_\alpha(w, v)$.

CASE 2.3. $\ell_v^{[\alpha]} \leq u_w^{[\alpha]} < \ell_w^{[\alpha]} \leq u_v^{[\alpha]}$. Then the intersection is

$$[\ell_v^{[\alpha]}, u_w^{[\alpha]}] \dot{\cup} [\ell_w^{[\alpha]}, u_v^{[\alpha]}] = \varphi_\alpha(v, w) \dot{\cup} \varphi_\alpha(w, v).$$

Clearly, if both $v^{[\alpha]}$ and $w^{[\alpha]}$ are intervals (CASE 3) (which are not disjoint) then their intersection is either $\varphi_\alpha(v, w)$ or $\varphi_\alpha(w, v)$. ◀

Pebble Transducers with Unary Output

Gaëtan Douéneau-Tabot ✉

IRIF, Université de Paris, France

Abstract

Bojańczyk recently initiated an intensive study of deterministic pebble transducers, which are two-way automata that can drop marks (named “pebbles”) on their input word, and produce an output word. They describe functions from words to words. Two natural restrictions of this definition have been investigated: marble transducers by Douéneau-Tabot et al., and comparison-free pebble transducers (that we rename here “blind transducers”) by Nguyen et al.

Here, we study the decidability of membership problems between the classes of functions computed by pebble, marble and blind transducers that produce a unary output. First, we show that pebble and marble transducers have the same expressive power when the outputs are unary (which is false over non-unary outputs). Then, we characterize 1-pebble transducers with unary output that describe a function computable by a blind transducer, and show that the membership problem is decidable. These results can be interpreted in terms of automated simplification of programs.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases polyregular functions, pebble transducers, marble transducers, streaming string transducers, factorization forests

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.40

Related Version *Full Version:* <https://arxiv.org/abs/2104.14019>

Funding *Gaëtan Douéneau-Tabot:* Funded by the French Ministry of Defence (DGA IP).

Acknowledgements The author is grateful to Olivier Carton for discussing about this work. He also thanks the reviewers for their helpful comments and remarks.

1 Introduction

Regular languages can be described by several models such as deterministic, non-deterministic, or two-way (the reading head can move in two directions) finite automata [12]. A natural extension consists in adding an output mechanism to finite automata. Such machines, called *transducers*, describe functions from words to words (or relations when non-deterministic) and provide a natural way to model simple programs that produce outputs. The particular model of a *two-way transducer* consists in a two-way automaton enhanced with an output function. It describes the class of *regular functions* which has been intensively studied for its fundamental properties: closure under composition [5], logical characterization by monadic second-order transductions [7], decidable equivalence problem [9], etc.

Pebble transducers and their variants. The model of k -pebble transducer can be defined as an inductive extension of two-way transducers. A 0-pebble transducer is just a two-way transducer. For $k \geq 1$, a k -pebble transducer \mathcal{T} is a two-way transducer that, when in a given configuration, can “call” an *external function* f , computed by some $(k-1)$ -pebble transducer. \mathcal{T} gives as argument to f its input word together with a mark, named “pebble”, on the position from which the call was performed, and uses the output of f within its own output.

The behavior of a 1-pebble transducer is depicted in Figure 1. Intuitively, a k -pebble transducer is some recursive program whose recursion depth is at most $k+1$. Equivalently, it can be seen as an iterative algorithm with “two-way for-loops”, such that the maximal depth of nested loops is $k+1$. A k -pebble transducer can only produce an output whose



© Gaëtan Douéneau-Tabot;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

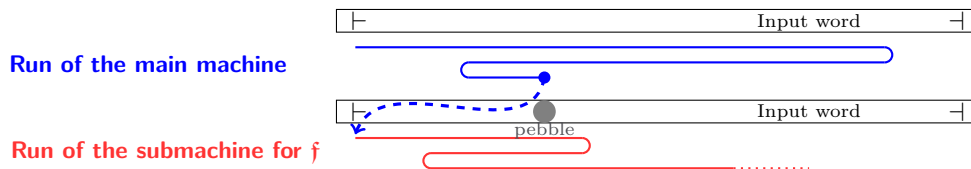
Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 40; pp. 40:1–40:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

length is polynomial in its input's length, more precisely $\mathcal{O}(n^{k+1})$ when n is the input's length (this is intuitive from the “nested loops” point of view). The functions computed by a k -pebble transducer for some $k \geq 0$ are thus called *polyregular functions* [3]. Several properties of polyregular functions have been investigated: closure under composition [3], logical characterization by monadic second-order interpretations [4], etc. The equivalence problem (given two machines, do they compute the same function?) is however still open.



■ **Figure 1** Behavior of a 1-pebble transducer.

Recently, two natural restrictions of pebble transducers have been introduced. First, the *k-marble transducers* of [6] only give as argument to their external function the prefix of the input word which ends in the calling position (see Figure 4). Second, the *k-blind transducers*¹ of [11] give the whole input word, but no pebble on the calling position (see Figure 3). The classes of functions they compute are strict subclasses of polyregular functions [6, 11].

Class membership problems. These various models of transducers raise several membership problems: given a function computed by a machine of model X , can it be computed by some machine of model Y ? When Y is a restriction of X , this problem reformulates as a program optimization question: given a “complex” algorithm in a class X , can we build an equivalent “simpler” one in class Y ? Thus it is of a foremost interest in practice.

Given a function f computed by an ℓ -pebble transducer, one can ask whether it is computable by a k -pebble transducer for a given $k < \ell$. The problem is decidable [10], and it turns out that a necessary and sufficient condition for this membership is that $|f(w)| = \mathcal{O}(|w|^{k+1})$. Using the “nested loops” interpretation of pebble transducers, it means that an output of size $\mathcal{O}(|w|^{k+1})$ can always be produced with at most $k+1$ nested loops. Similar results have been obtained in [6] and [11] for their variants, with the same conclusion: an output of size $\mathcal{O}(|w|^{k+1})$ can always be produced with depth at most $k+1$.

Contributions. In this paper, we study a different membership problem: can a function given by a k -pebble transducer be computed by a k -marble or k -blind transducer? It turns out to be a more difficult question, since there is no intuitive and machine-independent candidate for a membership condition (such as the size of the output). In general, membership problems for transducers are difficult, since contrary to regular languages, there is no “canonical” object known to represent a regular function. Hence, there can be several seemingly unrelated manners to produce the same function, and moving from one to another can be technical.

We focus on *transducers whose output alphabet is unary*, and our proof techniques are new. The first main result is that (when the outputs are unary) k -pebble transducers and k -marble transducers compute the same functions (one direction is obvious since k -marble is a restriction of k -pebble). The transformation is effective, but the way of producing the output must sometimes be completely modified (the transformation modifies the *origin semantics*, in the sense of [2]), which creates an additional difficulty. The correspondence fails as soon as the output is not over a unary alphabet, as detailed in Example 1.

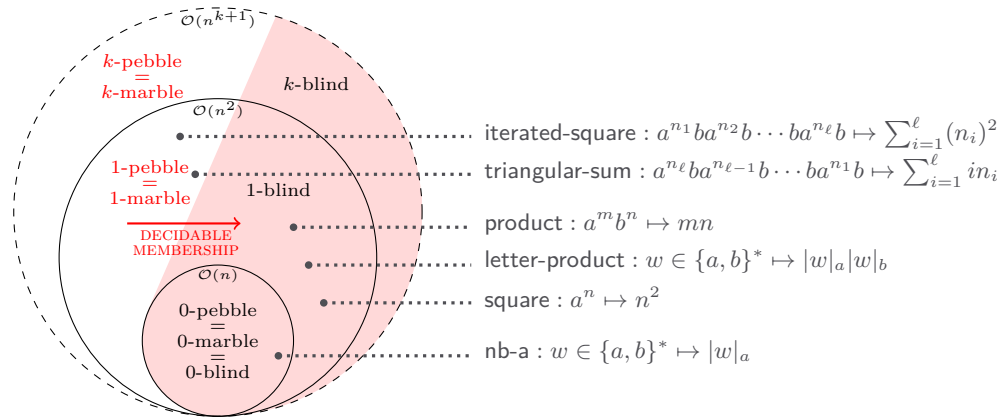
¹ The original terminology of [11] is *comparison-free pebble transducers*, but we strongly believe that the term “blind” is more adapted, since there are no pebbles in this model.

► **Example 1** ([6, 11]). The partial function $\{a, b\}^* \rightarrow \{a, b\}^*, a^m b^n \mapsto (b^n a)^m$ can be computed by a 1-pebble transducer, but not by a k -marble for any $k \geq 0$.

Since the equivalence problem is decidable for marble transducers, it follows from our result that it is also decidable for pebble transducers with unary output.

As a second main result, we show how to decide (when the outputs are unary) whether a function given by 1-pebble (\equiv 1-marble) transducer can be computed by a 1-blind transducer, or more generally by a k -blind transducer for some $k \geq 0$. The technical proof also gives a syntactical characterization of 1-marble transducers whose function verify this property: it describes a kind of “symmetry” in the production of the machine on its input. Furthermore, the conversion is effective when possible, but once more the manner of producing the output can be strongly modified. Our techniques heavily rely on the theory of *factorization forests*; this is, to our knowledge, the first time this notion is used for membership problems of transducers, and we believe this approach to be fruitful.

Our results are summarized in red in Figure 2. We also give some examples of functions (their outputs are non-negative integers, since we identify $\{a\}^*$ with \mathbb{N}).



■ **Figure 2** Classes of functions with unary output and results of this paper.

Outline. We first recall in Section 2 the definitions of k -pebble, k -marble and k -blind transducers, simplified for the case of unary outputs. In Section 3, we define the notions of k -pebble, k -marble and k -blind bimachines, and show their equivalence with the transducer models. Bimachines are easier to handle in the proofs, due to the fact that they avoid two-way moves. In Section 4, we show that k -marble and k -pebble transducers are equivalent. Finally, we solve in Section 5 the class membership problem from 1-pebble to 1-blind. Due to space constraints, several proofs are sketched in the main paper, and we focus on the most significant lemmas and characterizations.

2 Preliminaries

\mathbb{N} is the set of nonnegative integers. If $0 \leq i \leq j$, the set $[i:j]$ denotes $\{i, i+1, \dots, j\} \subseteq \mathbb{N}$ (empty if $j < i$). Capital letters A, B denote finite sets of letters (alphabets). The empty word is denoted ε . If $w \in A^*$, let $|w| \in \mathbb{N}$ be its length, and for $1 \leq i \leq |w|$ let $w[i]$ be its i -th letter. If $I = \{i_1 < \dots < i_\ell\} \subseteq \{1, \dots, |w|\}$, let $w[I] := w[i_1] \dots w[i_\ell]$. If $a \in A$, we denote by $|w|_a$ the number of letters a occurring in w . Given $A = \{a, \dots\}$, let $\underline{A} := \{\underline{a}, \dots\}$ be a

disjoint copy of A . For $1 \leq i \leq |w|$, we define $w \uparrow i := w[1:i-1] \underline{w[i]} w[i+1:|w|]$ as “ w in which position i is underlined”. We assume that the reader is familiar with the basics of automata theory, in particular the notion of two-way deterministic automaton.

Two-way transducers. A deterministic two-way transducer is a deterministic two-way automaton enhanced with the ability to produce outputs along its run. The class of functions described by these machines is known as *regular functions* [5, 7].

- **Definition 2.** A (deterministic) two-way transducer $(A, B, Q, q_0, F, \delta, \lambda)$ is:
- an input alphabet A and an output alphabet B ;
 - a finite set of states Q with an initial state $q_0 \in Q$ and a set $F \subseteq Q$ of final states;
 - a (partial) transition function $\delta : Q \times (A \uplus \{\vdash, \dashv\}) \rightarrow Q \times \{\triangleleft, \triangleright\}$;
 - a (partial) output function $\lambda : Q \times (A \uplus \{\vdash, \dashv\}) \rightarrow B^*$ with same domain as δ .

When given as input a word $w \in A^*$, the two-way transducer disposes of a read-only input tape containing $\vdash w \dashv$. The marks \vdash and \dashv are used to detect the borders of the tape, by convention we denote them as positions 0 and $|w|+1$ of w . Formally, a *configuration* over $\vdash w \dashv$ is a tuple (q, i) where $q \in Q$ is the current state and $0 \leq i \leq |w|+1$ is the position of the reading head. The *transition relation* \rightarrow is defined as follows. Given a configuration (q, i) , let $(q', \star) := \delta(q, w[i])$. Then $(q, i) \rightarrow (q', i')$ whenever either $\star = \triangleleft$ and $i' = i - 1$ (move left), or $\star = \triangleright$ and $i' = i + 1$ (move right), with $0 \leq i' \leq |w|+1$. A *run* is a sequence of configurations $(q_1, i_1) \rightarrow \dots \rightarrow (q_n, i_n)$. Accepting runs are those that begin in $(q_0, 0)$ and end in a configuration of the form $(q, |w|+1)$ with $q \in F$ (and it never visits such a configuration before). The function $f : A^* \rightarrow B^*$ computed by the machine is defined as follows. Let $w \in A^*$, if there exists an accepting run on $\vdash w \dashv$, then $f(w)$ is the concatenation of the $\lambda(q, w[i])$ along this unique run on $\vdash w \dashv$. To make f a total function, we let $f(w) := \varepsilon$ if there is no accepting run (the language of words having an accepting run in a two-way transducer is regular [12], hence the domain does not matter).

- **Example 3.** $\text{reverse} : A^* \rightarrow A^*$, $abac \mapsto caba$ can be computed by a two-way transducer.

From now on, the output alphabet of the machines will always be a singleton. Up to identifying $\{a\}^*$ and \mathbb{N} , we assume that $\lambda : Q \times (A \uplus \{\vdash, \dashv\}) \rightarrow \mathbb{N}$ and $f : A^* \rightarrow \mathbb{N}$.

External functions. We now extend the notion of output function λ : it will not give directly an integer, but performs a call to an *external function* which returns an integer. For pebbles, the output of the external functions depends on the input word and the current position.

- **Definition 4.** A two-way transducer with external pebble functions $(A, Q, q_0, F, \delta, \mathfrak{F}, \lambda)$ is:
- an input alphabet A ;
 - a finite set of states Q with an initial state $q_0 \in Q$ and a set $F \subseteq Q$ of final states;
 - a (partial) transition function $\delta : Q \times (A \uplus \{\vdash, \dashv\}) \rightarrow Q \times \{\triangleleft, \triangleright\}$;
 - a finite set \mathfrak{F} of external functions $\mathfrak{f} : (A \uplus \underline{A})^* \rightarrow \mathbb{N}$;
 - a (partial) output function $\lambda : Q \times (A \uplus \{\vdash, \dashv\}) \rightarrow \mathfrak{F}$ with same domain as δ .

Configurations (q, i) and runs of two-way transducers with external functions are defined as for classical two-way transducers. The function $f : A^* \rightarrow \mathbb{N}$ computed by the machine is defined as follows. Let $w \in A^*$ such that there exists an accepting run on $\vdash w \dashv$. If $\lambda(q, w[i]) = \mathfrak{f} \in \mathfrak{F}$, we let $\nu(q, i) := \mathfrak{f}(w \uparrow i)$, that is the result of \mathfrak{f} applied to w marked in i . Finally, $f(w)$ is defined as the sum of the $\nu(q, i)$ along this unique accepting run on $\vdash w \dashv$. We similarly set $f(w) = 0$ if there is no accepting run.

► Remark 5. If the external functions are constant, we exactly have a two-way transducer.

► Example 6. Let $a, b \in A$, $f_b : w \in (A \uplus \underline{A})^* \mapsto |w|_b$ and $f_0 : w \in (A \uplus \underline{A})^* \mapsto 0$. The two-way transducer with external pebble functions, which makes a single pass on its input and calls f_b if reading a and f_0 otherwise, computes **letter-product** : $w \in A \mapsto |w|_a|w|_b$.

We define two other models. Their definition is nearly the same, except that the external functions of \mathfrak{F} have type $A^* \rightarrow \mathbb{N}$ and $\nu(q, i)$ is defined in a slightly different way:

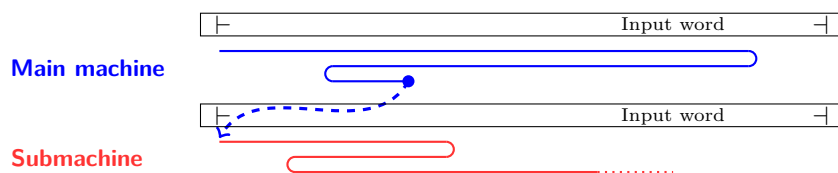
- in a *two-way transducer with external blind functions*, we define $\nu(q, i) := f(w)$. The external function is applied to w without marking the current position;
- in a *two-way transducer with external marble functions*, we define $\nu(q, i) := f(w[1:i])$. The external function is applied to the prefix of w stopping at the current position.

Pebble, blind and marble transducers. We now describe the transducer models using the formalism of external functions. These are not the original definitions from [3, 6, 11] (we are closer to the nested transducers of [10]), but the correspondence is straightforward, as soon as we know that pebble automata can only recognize regular languages.

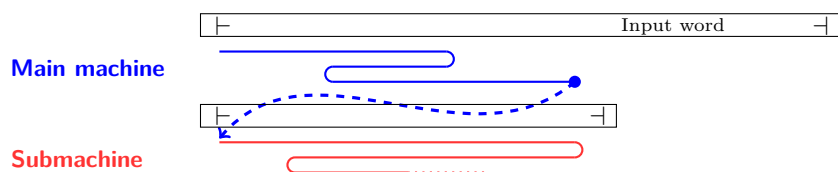
► Definition 7. For $k \geq 0$, a k -pebble (resp. k -blind, k -marble) transducer is:

- if $k = 0$, a two-way transducer;
- if $k \geq 1$, a two-way transducer with external pebble (resp. blind, marble) functions that are computed by $(k-1)$ -pebble (resp. $(k-1)$ -blind, $(k-1)$ -marble) transducers.

The intuitive behavior of a 1-pebble transducer is depicted in Figure 1 in Introduction. We draw in Figure 3 the behavior of a 1-blind transducer, which is the same except that the calling position is not marked for the machine computing the external function.



■ Figure 3 Behavior of a 1-blind transducer.



■ Figure 4 Behavior of a 1-marble transducer.

► Example 8. By restricting the functions f_b and f_0 of Example 6 to A^* , we see that **letter-product** : $w \in A^* \mapsto |w|_a|w|_b$ can be computed by a 1-blind transducer.

The intuitive behavior of a 1-marble transducer is depicted in Figure 4.

► Example 9. The function **letter-product** : $w \in A^* \mapsto |w|_a|w|_b$ can be computed by a 1-marble transducer as follows. Assume that $a \neq b$, let $f_a : w \mapsto |w|_a$ and $f_b : w \mapsto |w|_b$. The machine calls f_b when reading a and f_a when reading b . This way, each a is “counted” $|w|_b$ times (from the call of f_b starting in this a which computes all the b before it, plus each time it is seen in a call of f_a starting from some b after this a).

The strategy for computing letter-product is really different between Examples 8 and 9. This illustrates the difficulty to obtain a “canonical” form for a transduction.

3 From two-way transducers to bimachines

Since we consider a commutative output monoid, the order in which the production is performed does not matter. It is thus tempting to simplify a two-way transducer in a one-way machine which visits each position only once. This is exactly what we do with bimachines, with the subtlety that they are able to check regular properties of the prefix (resp. suffix) starting (resp. ending) in the current position. From now on, we consider only total functions of type $A^+ \rightarrow \mathbb{N}$ (the output on ε can be treated separately and does not matter).

- **Definition 10.** A bimachine with external pebble functions $(A, M, \mu, \mathfrak{F}, \lambda)$ consists of:
- an input alphabet A ;
 - a morphism into a finite monoid $\mu : A^* \rightarrow M$;
 - a finite set \mathfrak{F} of external functions $\mathfrak{f} : (A \uplus \underline{A})^+ \rightarrow \mathbb{N}$;
 - a total output function $\lambda : M \times A \times M \rightarrow \mathfrak{F}$.

Given $1 \leq i \leq |w|$ a position of $w \in A^*$, let $\mathfrak{f}_i := \lambda(\mu(w[1:i-1]), w[i], \mu(w[i+1:|w|])) \in \mathfrak{F}$. The bimachine defines a function $f : A^+ \rightarrow \mathbb{N}$ as follows:

$$f(w) := \sum_{1 \leq i \leq |w|} \mathfrak{f}_i(w \uparrow i).$$

As before, we can define *bimachines with external blind* (resp. *marble*) functions (in this case we have $\mathfrak{f}_i : A^+ \rightarrow \mathbb{N}$). We then let:

$$f(w) := \sum_{1 \leq i \leq |w|} \mathfrak{f}_i(w) \left(\text{resp. } f(w) := \sum_{1 \leq i \leq |w|} \mathfrak{f}_i(w[1:i]) \right).$$

As for two-way transducers, we define bimachines (without external functions) by setting $\lambda : M \times A \times M \rightarrow \mathbb{N}$. Equivalently, it corresponds to bimachines with external constant functions $\mathfrak{f} : w \mapsto n$. Going further, we define k -pebble bimachines by induction.

- **Definition 11.** For $k \geq 0$, a k -pebble (resp. k -blind, k -marble) bimachine is:
- if $k = 0$, a bimachine (without external functions);
 - if $k \geq 1$, a bimachine with external pebble (resp. blind, marble) functions which are computed by $(k-1)$ -pebble (resp. $(k-1)$ -blind, $(k-1)$ -marble) bimachines.

► **Example 12.** The function *triangular-sum* : $a^{n\ell} b a^{n\ell-1} b \dots b a^{n1} b \mapsto \sum_{i=1}^{\ell} i n_i$ can be computed by a 1-marble bimachine. It uses the singleton monoid $M = \{1_M\}$ and the morphism $\mu : a, b \mapsto 1_M$ in all its bimachines. The output function of the main bimachine is defined by $\lambda(1_M, a, 1_M) := \mathfrak{f}_a$ and $\lambda(1_M, b, 1_M) := \mathfrak{f}_b$. For computing $\mathfrak{f}_a : w \mapsto 0$ we use output $\lambda_{\mathfrak{f}_a}(1_M, a, 1_M) := 0$ and $\lambda_{\mathfrak{f}_a}(1_M, b, 1_M) := 0$, and for computing $\mathfrak{f}_b : w \mapsto |w|_a$ we use output $\lambda_{\mathfrak{f}_b}(1_M, a, 1_M) := 1$ and $\lambda_{\mathfrak{f}_b}(1_M, b, 1_M) := 0$.

Standard proof techniques allow to relate bimachines and transducers.

- **Proposition 13.** k -pebble (resp. k -blind, k -marble) bimachines and k -pebble (resp. k -blind, k -marble) transducers compute the same functions, and both conversions are effective.

Proof sketch. Both directions are treated by induction. From bimachines to transducers, we show that a bimachine with external pebble functions can be transformed in an equivalent two-way transducer with the same external pebble functions (we use a *lookaround* [7] to simulate μ). From transducers to bimachines, the induction step shows that a two-way transducer with external pebble functions can be transformed in an equivalent bimachine with external pebble functions, by adapting the classical reduction from two-way to one-way automata [12]. However the new external functions can be linear combinations of the former ones, since we produce “all at once” the results of several visits in a position. We only need to use a finite number of combinations, since in its accepting runs, a two-way transducer can only visit each position a bounded number of times. ◀

4 Equivalence between k -pebble and k -marble transducers

The main goal of this section is to show equivalence between k -pebble and k -marble transducers, over unary outputs. We shall use another model which is equivalent to marble transducers [6]: a *streaming string transducer (with unary output)*, which consists in a deterministic automaton with a finite set \mathfrak{X} of registers that store integers. At each letter read, the values of the registers are updated by doing a linear combination of their former values, whose coefficients depend on the current state of the automaton. In our definition we focus on the registers and forget about the states, which corresponds to a weighted automaton over the semiring $(\mathbb{N}, +, \times)$ (it is shown in [6] that both models are equivalent). The update is represented by a matrix from $\mathbb{N}^{\mathfrak{X} \times \mathfrak{X}}$, which is chosen depending on the letter read.

► **Definition 14.** A streaming string transducer (SST) $\mathcal{T} = (A, \mathfrak{X}, I, T, F)$ is:

- an input alphabet A and a finite set \mathfrak{X} of registers;
- an initial row vector $I \in \mathbb{N}^{\mathfrak{X}}$;
- a register update function $T : A \rightarrow \mathbb{N}^{\mathfrak{X} \times \mathfrak{X}}$;
- an output column vector $F \in \mathbb{N}^{\mathfrak{X}}$.

T can be extended as a monoid morphism from A^* to $(\mathbb{N}^{\mathfrak{X} \times \mathfrak{X}}, \times)$. Given $w \in A^*$, the vector $IT(w)$ intuitively describes the values of the registers after reading w . To define the function $f : A^* \rightarrow \mathbb{N}$ computed by \mathcal{T} , we combine these values by the output vector:

$$f(w) := IT(w)F.$$

► **Example 15.** The function *triangular-sum* : $a^{n_\ell} b a^{n_{\ell-1}} b \cdots b a^{n_1} b \mapsto \sum_{i=1}^{\ell} i n_i$ can be computed by an SST. We use two registers x, y and allow constants in the updates for more readability: x is initialized to 0 and updated $x \leftarrow x + 1$ on a and $x \leftarrow x$ on b , and y is initialized to 0 and updated $y \leftarrow y$ on a and $y \leftarrow y + x$ on b . Finally we output y .

We are now ready to state the main results of this section.

► **Theorem 16.** Given a k -pebble bimachine, one can build an equivalent SST.

The proof is done by induction on $k \geq 0$. Consider a bimachine whose external functions are computed by $(k-1)$ -pebble bimachines. By hypothesis, we can compute these functions by SSTs. The induction step is shown by Lemma 17, which uses new proof techniques.

► **Lemma 17.** Given a bimachine with external pebble functions computed by SSTs, one can build an equivalent SST (with no external functions).

Proof idea. Let \mathcal{T} be the SST computing an external function f . On input $w \in A^+$, the bimachine calls f on several positions $1 \leq i_1 < \dots < i_\ell \leq |w|$, which induces executions of \mathcal{T} on $w \uparrow i_1, \dots, w \uparrow i_\ell$. These executions are very similar: they only differ when reading the marked letter. Thus we build an SST which computes “simultaneously” all these executions, by keeping track of the sum of the values of the registers of \mathcal{T} along them. \blacktriangleleft

As a consequence of Theorem 16, we obtain equivalence between pebbles and marbles over unary outputs. The result is false over non-unary output alphabets [6, 11]. We also relate these functions with those computed by SST, assuming that the output is bounded by a polynomial in the input’s length.

► **Corollary 18.** *For all $k \geq 0$ and $f : A^* \rightarrow \mathbb{N}$, the following conditions are equivalent:*

1. f is computable by a k -pebble transducer;
2. f is computable by a k -marble transducer;
3. f is computable by an SST and $f(w) = \mathcal{O}(|w|^{k+1})$.

Furthermore the transformations are effective.

Proof. Clearly a k -pebble transducer can simulate a k -marble transducer, hence $2 \Rightarrow 1$. Let f be computed by a k -pebble transducer, we have $f(w) = \mathcal{O}(|w|^{k+1})$ and by Theorem 16 one can build an SST for f . Thus $1 \Rightarrow 3$. Finally $3 \Rightarrow 2$ is shown in [6]. \blacktriangleleft

Another important consequence is that we can decide equivalence of pebble transducers with unary output, since we can do so for marble transducers [6].

► **Corollary 19.** *One can decide if two pebble transducers compute the same function.*

This has been an open question since [3], and it is still open for generic output alphabets.

5 Deciding if 1-pebble is 1-blind

Since the equivalence between marbles and pebbles is established, we now compare 1-pebble (which are 1-marble) transducers with 1-blind transducers. It turns out that 1-pebble are strictly more expressive; the main goal of this section is to show Theorem 20.

► **Theorem 20 (Membership).** *One can decide if a function given by a 1-marble (or 1-pebble) transducer can be computed by a k -blind transducer for some $k \geq 0$. If this condition holds, one can build a 1-blind transducer which computes it.*

Let us fix a function $f : A^+ \rightarrow \mathbb{N}$ described by a 1-marble bimachine $\mathcal{T} = (A, M, \mu, \mathfrak{F}, \lambda)$. For $f \in \mathfrak{F}$, let $\mathcal{T}_f := (A, M, \mu, \lambda_f)$ be the bimachine which computes it. We enforce the morphism μ to be surjective (up to considering the co-restriction to its image) and the same in all machines (up to taking the product of all morphisms used). Our goal is to give a decidable condition on \mathcal{T} for f to be computable by a 1-blind transducer. For this purpose, we define the notion of *bitype*. Intuitively, it describes two disjoint factors in an input word, together with a finite abstraction of their “context”.

Let $\Lambda := 3|M|$ (it will be justified by Theorem 27).

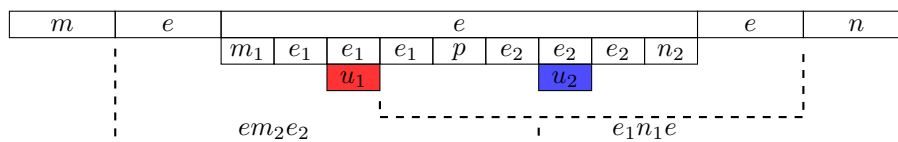
► **Definition 21.** *A bitype $\Phi := m \langle \mathbf{u}_1 \rangle m' \langle \mathbf{u}_2 \rangle m''$ consists in $m, m', m'' \in M$, $\mathbf{u}_1, \mathbf{u}_2 \in A^+$.*

We can define “the production performed in \mathbf{u}_1 by the calls from \mathbf{u}_2 ”, in Φ . For $1 \leq i \leq |\mathbf{u}_1|$ and $1 \leq j \leq |\mathbf{u}_2|$, let $\Phi(i, j) := \lambda_{f_j}(m\mu(\mathbf{u}_1[1:i-1]), \mathbf{u}_1[i], \mu(\mathbf{u}_1[i+1:|\mathbf{u}_1|]))m'\mu(\mathbf{u}_2[1:j])) \in \mathbb{N}$ where $f_j := \lambda(m\mu(\mathbf{u}_1)m'\mu(\mathbf{u}_2[1:j-1]), \mathbf{u}_2[j], \mu(\mathbf{u}_2[j+1:|\mathbf{u}_2|]))m''$). Then we set:

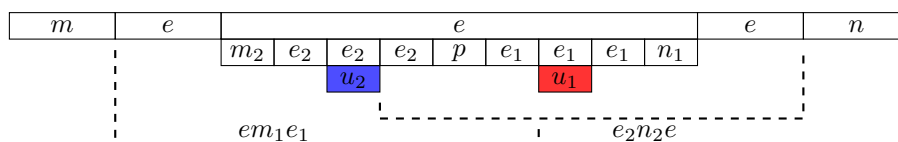
$$\text{prod}(m \langle \mathbf{u}_1 \rangle m' \langle \mathbf{u}_2 \rangle m'') := \sum_{\substack{1 \leq i \leq |\mathbf{u}_1| \\ 1 \leq j \leq |\mathbf{u}_2|}} \Phi(i, j) \in \mathbb{N}.$$

► **Definition 22.** The 1-marble bimachine \mathcal{T} is symmetrical whenever $\forall m, n, m_1, n_1, m_2, n_2 \in M$ and $u_1, u_2 \in A^+$ such that $|u_1|, |u_2| \leq 2^\Lambda$, $e_1 := \mu(u_1)$, $e_2 := \mu(u_2)$ and $e := m_1 e_1 n_1 = m_2 e_2 n_2$ are idempotents, there exists $K \geq 0$ such that $\forall p \in M$:

- if $m_1 e_1 p e_2 n_2 = e$, $e m_1 e_1 p e_2 = e m_2 e_2$ and $e_1 p e_2 n_2 e = e_1 n_1 e$, then $\text{prod}(m e m_1 e_1 \langle u_1 \rangle e_1 p e_2 \langle u_2 \rangle e_2 n_2 e n) = K$;
- if $m_2 e_2 p e_1 n_1 = e$, $e m_2 e_2 p e_1 = e m_1 e_1$ and $e_2 p e_1 n_1 e = e_2 n_2 e$, then $\text{prod}(m e m_2 e_2 \langle u_2 \rangle e_2 p e_1 \langle u_1 \rangle e_1 n_1 e n) = K$.



(a) Bitype $m e m_1 e_1 \langle u_1 \rangle e_1 p e_2 \langle u_2 \rangle e_2 n_2 e n$.



(b) Bitype $m e m_2 e_2 \langle u_2 \rangle e_2 p e_1 \langle u_1 \rangle e_1 n_1 e n$

■ **Figure 5** The bitypes used to define a symmetrical 1-marble bimachine.

Symmetry means that, under some idempotent conditions, $\text{prod}(m \langle u_1 \rangle m' \langle u_2 \rangle m'')$ only depends on $m, m'', m' \mu(u_2) m''$ and $m \mu(u_1) m'$, that are the “contexts” of u_1 and u_2 , but not on the element m' which separates them. The same holds if we swap u_1 and u_2 . The bitypes considered in Definition 22 are depicted in Figure 5, together with the equations they satisfy.

Symmetry is the decidable condition we are looking for, as shown in Theorem 23. Recall that f is the function computed by the 1-marble bimachine \mathcal{T} .

► **Theorem 23 (Characterization).** The following conditions are equivalent:

1. f is computable by a k -blind transducer for some $k \geq 0$;
2. f is computable by a 1-blind transducer;
3. \mathcal{T} is symmetrical.

Theorem 20 follows from Theorem 23, since it suffices to check whether the machine is symmetrical, which can be decided by ranging over all monoid elements (including idempotents) and words of length at most 2^Λ .

► **Example 24.** Let us show that the bimachine of Example 12 computing triangular-sum is not symmetrical. Let $u_1 := a$, $u_2 := b$, $m, n, m_1, n_1, m_2, n_2, p, e = 1_M$, $e_1 := \mu(u_1) = 1_M$ and $e_2 := \mu(u_2) = 1_M$. Then $\text{prod}(m e m_1 e_1 \langle u_1 \rangle e_1 p e_2 \langle u_2 \rangle e_2 n_2 e n) = \text{prod}(1_M \langle a \rangle 1_M \langle b \rangle 1_M) = 1$ and $\text{prod}(m e m_2 e_2 \langle u_2 \rangle e_2 p e_1 \langle u_1 \rangle e_1 n_1 e n) = \text{prod}(1_M \langle b \rangle 1_M \langle a \rangle 1_M) = 0$. Furthermore the equations of Definition 22 hold, thus triangular-sum is not computable by a k -blind bimachine.

Lemma 25 shows $1 \Rightarrow 3$ in Theorem 23. It allows to show that some function cannot be computed by a k -marble transducer. Its proof is technical; a coarse intuition is that a 1-blind bimachine which makes a production on u_1 when called from u_2 cannot see the monoid element m' between u_1 and u_2 (since u_2 is not marked, its position is “forgotten”).

► **Lemma 25.** If f is computable by a k -blind bimachine, then \mathcal{T} is symmetrical.

Since $2 \Rightarrow 1$ in Theorem 23 is obvious, it remains to show that if \mathcal{T} is symmetrical, then f is effectively computable by a 1-blind bimachine. This is the goal of the two following subsections. The main tool for the proof is the notion of factorization forest: using Lemma 36, it allows us to compute the function f without directly referring to a machine.

5.1 Factorization forests

Recall that $\mu : A^+ \rightarrow M$ is a fixed monoid morphism. A *factorization forest* [1] of $w \in A^+$ is an unranked tree structure which decomposes w following the image of its factors by μ .

- **Definition 26** ([13, 1]). A factorization (forest) of $w \in A^+$ is a tree defined as follows:
 - if $w = a \in A$, it is a leaf a ;
 - if $|w| \geq 2$, then $(\mathcal{F}_1) \cdots (\mathcal{F}_n)$ is a factorization of w if each \mathcal{F}_i is a factorization of some $w_i \in A^+$ such that $w = w_1 \cdots w_n$, and either:
 - $n = 2$: the root is a binary node;
 - or $n \geq 3$ and $\mu(w_1) = \cdots = \mu(w_n)$ is an idempotent: the root is an idempotent node.

The set of factorizations over w is denoted $\text{Fact}(w)$. Recall that $\Lambda = 3|M|$.

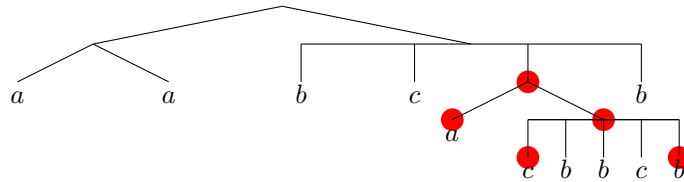
- **Theorem 27** ([13, 1]). For all $w \in A^+$, there is $\mathcal{F} \in \text{Fact}(w)$ of height at most Λ .

Let $\widehat{A} := A \uplus \{(\cdot, \cdot)\}$. We have defined $\text{Fact}(w)$ as a set of tree structures, but we can assume that $\text{Fact}(w) \subseteq \widehat{A}^+$. Indeed, in Definition 26, a factorization of w can also be seen as “the word w with parentheses”. There exists a rational function which computes factorizations, under this formalism. We reformulate this statement in Proposition 28 using a two-way transducer (which, exceptionally in this paper, has a non-unary output alphabet \widehat{A}).

- **Proposition 28** (Folklore). One can build a two-way transducer which computes a function $A^+ \rightarrow \widehat{A}^+, w \mapsto \mathcal{F} \in \text{Fact}(w)$ for some \mathcal{F} of height at most Λ .

We denote by $\text{Nodes}(\mathcal{F})$ the set of (idempotent or binary) nodes of \mathcal{F} . In order to simplify the statements, we identify a node with the subtree rooted in this node. Thus $\text{Nodes}(\mathcal{F})$ can also be seen as the set of subtrees of \mathcal{F} , and $\mathcal{F} \in \text{Nodes}(\mathcal{F})$. We shall use the standard tree vocabulary of “height” (a leaf is a tree of height 1), “parent node”, “descendant” and “ancestor” (defined in a non-strict way: a node is itself one of its ancestors), “branch”, etc.

- **Example 29.** Let $A = \{a, b, c\}$, $M = \{1_M, 2_M, 3_M\}$ with $2_M^2 = 1_M$, 3_M absorbing, $\mu(a) := 2_M$ and $\mu(b) := \mu(c) := 3_M$. Then $\mathcal{F} := (aa)(bc(a(cbbcb)))b \in \text{Fact}(aacacbbcb)$ (we dropped the parens around single letters for more readability) is depicted in Figure 6. Idempotent nodes are drawn using a horizontal line.



■ **Figure 6** The factorization $(aa)(bc(a(cbbcb)))b$ of $aacacbbcb$.

We define $\text{Iterable-nodes}(\mathcal{F}) \subseteq \text{Nodes}(\mathcal{F})$ as the set of nodes which are the middle child of an idempotent node. Intuitively, such nodes can be copied without modifying their “context”.

► **Definition 30.** Let $\mathcal{F} \in \text{Fact}(w)$, we define the set of iterable nodes of \mathcal{F} by induction:

- if $\mathcal{F} = a \in A$ is a leaf, $\text{Iterable-nodes}(\mathcal{F}) := \emptyset$;
- if $\mathcal{F} = (\mathcal{F}_1) \cdots (\mathcal{F}_n)$ is a binary or idempotent node, then:

$$\text{Iterable-nodes}(\mathcal{F}) := \{\mathcal{F}_i : 2 \leq i \leq n-1\} \bigsqcup_{1 \leq i \leq n} \text{Iterable-nodes}(\mathcal{F}_i).$$

On the contrary, we now define sets of nodes which cannot be duplicated individually.

► **Definition 31.** Let $\mathcal{F} \in \text{Fact}(w)$, we define the dependency of \mathcal{F} as follows:

- if $\mathcal{F} = a \in A$ is a leaf, then $\text{Dep}(\mathcal{F}) := \{a\}$;
- if $\mathcal{F} = (\mathcal{F}_1) \cdots (\mathcal{F}_n)$ is binary or idempotent, then $\text{Dep}(\mathcal{F}) := \{\mathcal{F}\} \cup \text{Dep}(\mathcal{F}_1) \cup \text{Dep}(\mathcal{F}_n)$.

Intuitively, $\text{Dep}(\mathcal{F}) \subseteq \text{Nodes}(\mathcal{F})$ contains all the nodes of \mathcal{F} except those which are descendant of a middle child. If $\mathcal{I} \in \text{Nodes}(\mathcal{F})$, we consider $\text{Dep}(\mathcal{I}) \subseteq \text{Nodes}(\mathcal{I})$ as a subset of $\text{Nodes}(\mathcal{F})$. We then define the frontier of \mathcal{I} , denoted $\text{Fr}_{\mathcal{F}}(\mathcal{I}) \subseteq \{1, \dots, |w|\}$ as the set of positions of w which belong to $\text{Dep}(\mathcal{I})$ (when seen as leaves of \mathcal{F}).

► **Example 32.** In Figure 6, the top-most red node \mathcal{I} is iterable. Furthermore $\text{Dep}(\mathcal{I})$ is the set of red nodes, $\text{Fr}_{\mathcal{F}}(\mathcal{I}) = \{5, 6, 10\}$ and $w[\text{Fr}_{\mathcal{F}}(\mathcal{I})] = acb$.

The relationship between iterable nodes and dependencies is detailed below. We denote by $\text{Part}(\mathcal{F}) := \text{Iterable-nodes}(\mathcal{F}) \uplus \{\mathcal{F}\}$, the set of iterable nodes plus the root.

► **Lemma 33.** Let $\mathcal{F} \in \text{Fact}(w)$, then $\{\text{Dep}(\mathcal{I}) : \mathcal{I} \in \text{Part}(\mathcal{F})\}$ is a partition of $\text{Nodes}(\mathcal{F})$; and $\{\text{Fr}_{\mathcal{F}}(\mathcal{I}) : \mathcal{I} \in \text{Part}(\mathcal{F})\}$ is a partition of $\{1, \dots, |w|\}$.

We define $\text{prod}(i, j)$ in w as “the production performed in i when called from j ”.

► **Definition 34.** Let $w \in A^+$ and $1 \leq i \leq j \leq |w|$ two positions of w . We define $\text{prod}(i, j) \in \mathbb{N}$ as $\lambda_{\mathfrak{f}_j}(\mu(w[1:i-1]), w[i], \mu(w[i+1:j]))$, where $\mathfrak{f}_j := \lambda(\mu(w[1:j-1]), w[j], \mu(w[j+1:|w|]))$.

We extend this definition to pairs of nodes: given $\mathcal{I}, \mathcal{J} \in \text{Nodes}(\mathcal{F})$, we define $\text{prod}(\mathcal{I}, \mathcal{J})$ “the sum of all productions performed in the frontier of \mathcal{I} , when called from the frontier of \mathcal{J} ” as follows (we have to ensure that the calling positions are “on the right”).

► **Definition 35.** Let $w \in A^+$, $\mathcal{F} \in \text{Fact}(w)$ and $\mathcal{I}, \mathcal{J} \in \text{Nodes}(\mathcal{F})$. We define:

$$\text{prod}(\mathcal{I}, \mathcal{J}) := \sum_{\substack{i \in \text{Fr}_{\mathcal{F}}(\mathcal{I}) \\ j \in \text{Fr}_{\mathcal{F}}(\mathcal{J}) \\ i \leq j}} \text{prod}(i, j) \in \mathbb{N}.$$

If \mathcal{I} is an ancestor of \mathcal{J} (or the converse) then $\text{Fr}_{\mathcal{F}}(\mathcal{I})$ and $\text{Fr}_{\mathcal{F}}(\mathcal{J})$ are interleaved, hence we can have both $\text{prod}(\mathcal{I}, \mathcal{J}) \neq 0$ and $\text{prod}(\mathcal{J}, \mathcal{I}) \neq 0$. However, if \mathcal{I} and \mathcal{J} are not on the same branch, we have either $\text{prod}(\mathcal{I}, \mathcal{J}) = 0$ or $\text{prod}(\mathcal{J}, \mathcal{I}) = 0$.

Applying Lemma 33, it is not hard to compute $f(w)$ using the $\text{prod}(\mathcal{I}, \mathcal{J})$.

► **Lemma 36.** Let $w \in A^+$, $\mathcal{F} \in \text{Fact}(w)$. Then:

$$f(w) = \sum_{\mathcal{I}, \mathcal{J} \in \text{Part}(\mathcal{F})} \text{prod}(\mathcal{I}, \mathcal{J}).$$

5.2 Typology of pairs of nodes

We intend to compute (if possible) f using a 1-blind transducer. Following Lemma 36, it is enough to consider the productions performed on the pairs of nodes of a factorization. For this study, we split the pairs depending on their relative position in the tree.

40:12 Pebble Transducers with Unary Output

Pairs separated by the frontier of the root. The frontier of the root $\text{Fr}_{\mathcal{F}}(\mathcal{F})$ plays a very specific role with respect to blind transducers. Indeed, over factorizations of height at most Λ , the size of the frontier is bounded, hence it splits the word in a bounded number of distinguishable “blocks”. Formally, we define the notion of *basis*.

► **Definition 37.** *An idempotent node is a basis if it belongs to the dependency of the root.*

The following result is shown by induction.

► **Lemma 38.** *Let $w \in A^+$ and $\mathcal{F} \in \text{Fact}(w)$. Given $\mathcal{I} \in \text{Iterable-nodes}(\mathcal{F})$, there exists a unique basis, denoted $\text{basis}_{\mathcal{F}}(\mathcal{I})$, such that \mathcal{I} is the descendant of a middle child of $\text{basis}_{\mathcal{F}}(\mathcal{I})$.*

► **Definition 39.** *Given $w \in A^+$ and $\mathcal{F} \in \text{Fact}(w)$, we define $D(\mathcal{F}) \subseteq \text{Part}(\mathcal{F}) \times \text{Part}(\mathcal{F})$ by:*

$$D(\mathcal{F}) := \{(\mathcal{I}, \mathcal{J}) : \mathcal{I}, \mathcal{J} \in \text{Iterable-nodes}(\mathcal{F}) \text{ and } \text{basis}_{\mathcal{F}}(\mathcal{I}) \neq \text{basis}_{\mathcal{F}}(\mathcal{J})\}.$$

Intuitively $\text{basis}_{\mathcal{F}}(\mathcal{I}) \neq \text{basis}_{\mathcal{F}}(\mathcal{J})$ means that $\text{Fr}_{\mathcal{F}}(\mathcal{I})$ and $\text{Fr}_{\mathcal{F}}(\mathcal{J})$ belong to two different “blocks” of the input. Lemma 40 is shown by building a 1-blind bimachine which visits successively each basis \mathcal{B} , and for each iterable \mathcal{J} such that $\text{basis}_{\mathcal{F}}(\mathcal{J}) = \mathcal{B}$, calls a submachine which visits the \mathcal{I} such that $\text{basis}_{\mathcal{F}}(\mathcal{I}) \neq \mathcal{B}$ and produces $\text{prod}(\mathcal{I}, \mathcal{J})$. The key element for doing this operation without pebbles is that the number of bases is bounded.

► **Lemma 40.** *One can build a 1-blind bimachine computing:*

$$f_D : (\widehat{A})^+ \rightarrow \mathbb{N}, \mathcal{F} \mapsto \begin{cases} \sum_{(\mathcal{I}, \mathcal{J}) \in D(\mathcal{F})} \text{prod}(\mathcal{I}, \mathcal{J}) & \text{if } \mathcal{F} \text{ factorization of height at most } \Lambda; \\ 0 & \text{otherwise.} \end{cases}$$

Linked pairs. Let $U(\mathcal{F}) := \text{Part}(\mathcal{F}) \times \text{Part}(\mathcal{F}) \setminus D(\mathcal{F})$, it corresponds to the pairs of $\text{Iterable-nodes}(\mathcal{F})$ which have the same basis, plus all the pairs $(\mathcal{F}, \mathcal{I})$ and $(\mathcal{I}, \mathcal{F})$ for $\mathcal{I} \in \text{Part}(\mathcal{F})$. We now study the pairs of $U(\mathcal{F})$ which are “linked”, in the sense that one node is (nearly) the ancestor of the other.

► **Definition 41.** *Let $w \in A^+$, $\mathcal{F} \in \text{Fact}(w)$. Let $L(\mathcal{F})$ be the set of all $(\mathcal{I}, \mathcal{J}) \in U(\mathcal{F})$ such that \mathcal{I} (or \mathcal{J}) is either the ancestor of, or the right/left sibling of an ancestor of \mathcal{J} (or \mathcal{I}).*

In particular, we have $(\mathcal{F}, \mathcal{F}), (\mathcal{I}, \mathcal{F}), (\mathcal{F}, \mathcal{I}), (\mathcal{I}, \mathcal{I}) \in L(\mathcal{F})$ for all $\mathcal{I} \in \text{Part}(\mathcal{F})$. If \mathcal{F} has height at most Λ , there are at most 3Λ nodes which are either an ancestor or the right/left sibling of an ancestor of \mathcal{I} . Lemma 42 follows from this boundedness.

► **Lemma 42.** *One can build a 0-blind bimachine computing:*

$$f_L : (\widehat{A})^+ \rightarrow \mathbb{N}, \mathcal{F} \mapsto \begin{cases} \sum_{(\mathcal{I}, \mathcal{J}) \in L(\mathcal{F})} \text{prod}(\mathcal{I}, \mathcal{J}) & \text{if } \mathcal{F} \text{ factorization of height at most } \Lambda; \\ 0 & \text{otherwise.} \end{cases}$$

Independent nodes. The remaining sum is the most interesting, since it is the only case where we use the assumption that \mathcal{T} to be symmetrical (and this assumption is crucial). Let $\mathcal{F} \in \text{Fact}(w)$, we define the set $I(\mathcal{F}) := U(\mathcal{F}) \setminus L(\mathcal{F})$. It contains the pairs $(\mathcal{I}, \mathcal{J})$ of iterable nodes such that $\text{basis}_{\mathcal{F}}(\mathcal{I}) = \text{basis}_{\mathcal{F}}(\mathcal{J})$ (i.e. they descend from a common “big” idempotent), and \mathcal{I} (or \mathcal{J}) is not an ancestor of \mathcal{J} (or \mathcal{I}), nor the left or right sibling of its ancestor.

► **Lemma 43.** *If \mathcal{T} is symmetrical, one can build a 1-blind bimachine computing:*

$$f_I : (\widehat{A})^+ \rightarrow \mathbb{N}, \mathcal{F} \mapsto \begin{cases} \sum_{(\mathcal{I}, \mathcal{J}) \in I(\mathcal{F})} \text{prod}(\mathcal{I}, \mathcal{J}) & \text{if } \mathcal{F} \text{ factorization of height at most } \Lambda; \\ 0 & \text{otherwise.} \end{cases}$$

Proof idea. We define $\text{type}_{\mathcal{F}}(\mathcal{I})$ for $\mathcal{I} \in \text{Iterable-nodes}(\mathcal{F})$ as a bounded abstraction of \mathcal{I} which describes the frontier and the location of \mathcal{I} in \mathcal{F} and in $\text{basis}_{\mathcal{F}}(\mathcal{I})$. Using symmetry, we show that for $(\mathcal{I}, \mathcal{J}) \in I(\mathcal{F})$, $\text{prod}(\mathcal{I}, \mathcal{J})$ only depends on $\text{type}_{\mathcal{F}}(\mathcal{I})$ and $\text{type}_{\mathcal{F}}(\mathcal{J})$, but not on their relative positions. Hence we build a 1-blind bimachine, whose main bimachine ranges over all possible \mathcal{J} and computes $\text{type}_{\mathcal{F}}(\mathcal{J})$, and whose submachines range over all possible \mathcal{I} (a special treatment has to be done to avoid \mathcal{I} such that $(\mathcal{I}, \mathcal{J}) \in L(\mathcal{F})$), compute $\text{type}_{\mathcal{F}}(\mathcal{I})$ and output $\text{prod}(\mathcal{I}, \mathcal{J})$. The submachines do not need to “see” \mathcal{J} . ◀

We finally show $3 \Rightarrow 2$ in Theorem 23. Given $w \in A^+$ we first compute \mathcal{F} of height at most Λ by Proposition 28. Then we use the machines from Lemmas 40, 42 and 43 and build a 1-blind transducer computing the sum of their outputs. The original function can be recovered since 1-blind transducers are closed under composition with two-way [11].

6 Conclusion and outlook

As a conclusion, we discuss future work. This paper introduces new proof techniques, in particular the use of factorization forests to study the productions of transducers. We believe that these techniques give a step towards other membership problems concerning pebble transducers. Among them, let us mention the membership problem from k -pebble to k -blind, at first over unary alphabets. Similarly, the membership from k -pebble to k -marble over non-unary alphabets is worth being studied (the answer seems to rely on combinatorial properties of the output, since unary outputs can always be produced using marbles).

References

- 1 Mikołaj Bojańczyk. Factorization forests. In *International Conference on Developments in Language Theory*, pages 1–17. Springer, 2009.
- 2 Mikołaj Bojańczyk. Transducers with origin information. In *International Colloquium on Automata, Languages, and Programming*, pages 26–37. Springer, 2014.
- 3 Mikołaj Bojańczyk. Polyregular functions. *arXiv preprint*, 2018. [arXiv:1810.08760](https://arxiv.org/abs/1810.08760).
- 4 Mikołaj Bojańczyk, Sandra Kiefer, and Nathan Lhote. String-to-string interpretations with polynomial-size output. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, pages 106:1–106:14, 2019.
- 5 Michal P Chytil and Vojtěch Jákł. Serial composition of 2-way finite-state transducers and simple programs on strings. In *4th International Colloquium on Automata, Languages, and Programming, ICALP 1977*, pages 135–147. Springer, 1977.
- 6 Gaëtan Douéneau-Tabot, Emmanuel Filiot, and Paul Gastin. Register transducers are marble transducers. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24–28, 2020, Prague, Czech Republic*, 2020.
- 7 Joost Engelfriet and Hendrik Jan Hoogetboom. MSO definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic (TOCL)*, 2(2):216–254, 2001.
- 8 Emmanuel Filiot and Pierre-Alain Reynier. Copyful streaming string transducers. In *International Workshop on Reachability Problems*, pages 75–86. Springer, 2017.
- 9 Eitan M Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM Journal on Computing*, 11(3):448–452, 1982.

- 10 Nathan Lhote. Pebble minimization of polyregular functions. In *35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*. IEEE, 2020.
- 11 Lê Thành Dung Nguyễn, Camille Noûs, and Pierre Pradic. Comparison-free polyregular functions. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, 2021.
- 12 John C Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.
- 13 Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.

A Proof of Lemma 17

We show that given a bimachine with external pebble functions, which are computed by SSTs, one can build an equivalent SST.

A.1 SST with lookahead

We first define a variant of SST with the same expressive power. Intuitively, this model is similar to bimachines, in the sense that the register update not only depends on the current letter, but also on a finite abstraction of the prefix and suffix.

► **Definition 44.** An SST with lookahead $\mathcal{T} = (A, \mathfrak{X}, M, \mu, I, \lambda, F)$ is:

- an input alphabet A and a finite set \mathfrak{X} of registers;
- a morphism into a finite monoid $\mu : A^* \rightarrow M$;
- an initial row vector $I \in \mathbb{N}^{\mathfrak{X}}$;
- a register update function $\lambda : M \times A \times M \rightarrow \mathbb{N}^{\mathfrak{X} \times \mathfrak{X}}$;
- an output column vector $F \in \mathbb{N}^{\mathfrak{X}}$.

Let us define its semantics. Intuitively, in position i of $w \in A^+$, we perform the register update $\lambda(\mu(w[1:i-1]), w[i], \mu(w[i+1:|w|]))$. Formally, for $0 \leq i \leq |w|$, we define $\mathcal{T}^{w,i} \in \mathbb{N}^{\mathfrak{X}}$ (“the values of the registers after reading $w[1:i]$ ”²) as follows:

- $\mathcal{T}^{w,0} := I$;
- for $i \geq 1$, $\mathcal{T}^{w,i} := \mathcal{T}^{w,i-1} \times \lambda(\mu(w[1:i-1]), w[i], \mu(w[i+1:|w|]))$.

To define the function $f : A^+ \rightarrow \mathbb{N}$ computed by \mathcal{T} , we combine the final values by the output vector:

$$f(w) := \mathcal{T}^{w,|w|} F.$$

It is known that SST with lookahead are equivalent to SST (the proof is roughly a “determinisation” procedure for eliminating the rightmost argument of λ , and an encoding of the monoid in the registers for eliminating the leftmost argument).

► **Lemma 45 ([8]).** Given an SST with lookahead, we can build an equivalent SST.

Hence, it is sufficient to build an SST with lookahead.

² Due to the fact that λ looks “on the right”, $\mathcal{T}^{w,i}$ depends on the whole w and not only on $w[1:i]$.

A.2 Main proof of Lemma 17

Let $\mathcal{T} = (A, M, \mu, \lambda, \mathfrak{F})$ be the bimachine with external pebble functions. Each $f \in \mathfrak{F}$ is computed by an SST $\mathcal{T}_f := (A \uplus \underline{A}, \mathfrak{X}_f, I_f, T_f, F_f)$.

► **Example 46** (Running example). Let $\mathcal{T} = (A, M, \mu, \lambda, \{f\})$ with $M = \{1_M\}$ singleton and $\lambda(1_M, a, 1_M) = f$ for $a \in A$. Let \mathcal{T}_f have two registers x, y with x initialized to 1 and y initialized to 0. When reading $a \in A \uplus \underline{A}$ it performs $x \leftarrow x, y \leftarrow y + x$. Finally it outputs y . Then $f(w) = |w|$ and \mathcal{T} computes $f : w \mapsto |w|^2$.

► **Definition 47.** Let $w \in A^*$ and $f \in \mathfrak{F}$. We define $\text{Calls}(w, f)$ as the set of positions of w in which f is called, that is $\{1 \leq j \leq |w| : \lambda(\mu(w[1:j-1]), w[j], \mu(w[j+1:|w|])) = f\}$.

For $1 \leq j \leq i \leq |w|$, $I_f T_f(w[1:i] \uparrow j)$ corresponds to the value of the registers of \mathcal{T}_f in position i when the call to f is performed from position j .

▷ **Claim 48.** For all $w \in A^*$, the following holds:

$$f(w) = \sum_{f \in \mathfrak{F}} \sum_{\substack{1 \leq j \leq |w| \\ j \in \text{Calls}(w, f)}} I_f T_f(w \uparrow j) F_f$$

Proof. By definition of external pebble functions we have:

$$f(w) := \sum_{1 \leq j \leq |w|} f_j(w \uparrow j)$$

where $f_j := \lambda(\mu(w[1:j-1]), w[j], \mu(w[j+1:|w|]))$ is “the external function called in j ”. Hence by partitioning the sum depending on the external functions it follows:

$$f(w) := \sum_{f \in \mathfrak{F}} \sum_{\substack{1 \leq j \leq |w| \\ j \in \text{Calls}(w, f)}} f(w \uparrow j)$$

And finally we note that $f(w \uparrow j) = I_f T_f(w \uparrow j) F_f$. ◁

Idea of the construction. Let us fix an external function f . Following Claim 48, we want to build an SST with lookahead \mathcal{U} which computes the values of the vector:

$$\sum_{\substack{1 \leq j \leq |w| \\ j \in \text{Calls}(w, f)}} I_f T_f(w \uparrow j) \in \mathbb{N}^{\mathfrak{X}}.$$

For this, it will keep track when in position i of the values of:

$$\sum_{\substack{1 \leq j \leq i \\ j \in \text{Calls}(w, f)}} I_f T_f(w[1:i] \uparrow j) \in \mathbb{N}^{\mathfrak{X}}$$

and these values will be updated when going from i to $i + 1$.

► **Example 49** (Running example). We have $I_f T_f(w[1:i] \uparrow j)(x) = 1$ and $I_f T_f(w[1:i] \uparrow j)(y) = i$. Hence $\sum_{\substack{1 \leq j \leq i \\ j \in \text{Calls}(w, f)}} I_f T_f(w[1:i] \uparrow j)(x) = i$ and $\sum_{\substack{1 \leq j \leq i \\ j \in \text{Calls}(w, f)}} I_f T_f(w[1:i] \uparrow j)(y) = i^2$.

40:16 Pebble Transducers with Unary Output

Formal construction. Let $\mathcal{U} = (A, \mathfrak{X}, M, \mu, I, \kappa, F)$ be an SST with lookahead with:

- the set $\mathfrak{X} := \bigsqcup_{\mathfrak{f} \in \mathfrak{F}} \{\text{Sum}_x : x \in \mathfrak{X}_{\mathfrak{f}}\} \sqcup \{\text{Old}_x : x \in \mathfrak{X}_{\mathfrak{f}}\}$ of registers;
- the morphism $\mu : A^* \rightarrow M$ used in \mathcal{T} ;
- an initial column vector $I \in \mathbb{N}^{\mathfrak{X}}$ such that for all $\mathfrak{f} \in \mathfrak{F}$ and $x \in \mathfrak{X}_{\mathfrak{f}}$:
 - $I(\text{Sum}_x) = 0$;
 - $I(\text{Old}_x) = I_{\mathfrak{f}}(x)$;
- the update $\kappa : M \times A \times M \rightarrow \mathbb{N}^{\mathfrak{X} \times \mathfrak{X}}$ as follows. Let $(m, a, n) \in M \times A \times N$ and $\mathfrak{f} := \lambda(m, a, n)$. Then $\kappa(m, a, n)$ performs the following updates:
 - for all $\mathfrak{g} \neq \mathfrak{f}$ and $x \in \mathfrak{X}_{\mathfrak{g}}$:
 - * $\text{Sum}_x \leftarrow \sum_{y \in \mathfrak{X}_{\mathfrak{g}}} \alpha_y \text{Sum}_y$;
 - * $\text{Old}_x \leftarrow \sum_{y \in \mathfrak{X}_{\mathfrak{g}}} \alpha_y \text{Old}_y$;
 where $x \leftarrow \sum_{y \in \mathfrak{X}_{\mathfrak{g}}} \alpha_y y$ is the update performed by $\mathcal{T}_{\mathfrak{g}}$ when reading a ;
 - for all $x \in \mathfrak{X}_{\mathfrak{f}}$:
 - * $\text{Sum}_x \leftarrow \sum_{y \in \mathfrak{X}_{\mathfrak{f}}} \alpha_y \text{Sum}_y + \sum_{y \in \mathfrak{X}_{\mathfrak{f}}} \beta_y \text{Old}_y$;
 - * $\text{Old}_x \leftarrow \sum_{y \in \mathfrak{X}_{\mathfrak{f}}} \alpha_y \text{Old}_y$;
 where $x \leftarrow \sum_{y \in \mathfrak{X}_{\mathfrak{f}}} \alpha_y y$ is the update performed by $\mathcal{T}_{\mathfrak{f}}$ when reading a ;
 and $x \leftarrow \sum_{y \in \mathfrak{X}_{\mathfrak{f}}} \beta_y y$ is the update performed by $\mathcal{T}_{\mathfrak{f}}$ when reading a .
 Intuitively, the sum with the β_y corresponds to what is “added” by the new call to \mathfrak{f} .
- the output line vector $F \in \mathbb{N}^{\mathfrak{X}}$ such that for all $\mathfrak{f} \in \mathfrak{F}$ and $x \in \mathfrak{X}_{\mathfrak{f}}$:
 - $F(\text{Sum}_x) = F_{\mathfrak{f}}(x)$;
 - $F(\text{Old}_x) = 0$.

► **Example 50** (Running example). \mathcal{U} performs the following updates:

- $\text{Old}_x \leftarrow \text{Old}_x$, $\text{Old}_y \leftarrow \text{Old}_y + \text{Old}_x$;
- $\text{Sum}_x \leftarrow \text{Sum}_x + \text{Old}_x$, $\text{Sum}_y \leftarrow \text{Sum}_y + \text{Sum}_x + \text{Old}_y + \text{Old}_x$.

We can check that $\mathcal{U}^{w,i}(\text{Old}_x) = 1$, $\mathcal{U}^{w,i}(\text{Old}_y) = i$ and $\mathcal{U}^{w,i}(\text{Sum}_x) = i$, $\mathcal{U}^{w,i}(\text{Sum}_y) = i^2$.

Correctness of the construction. As the registers Old_x for $x \in \mathfrak{X}_{\mathfrak{f}}$ are updated following the updates of $\mathcal{T}_{\mathfrak{f}}$, it follows immediately that:

▷ **Claim 51.** Given $x \in \mathfrak{X}_{\mathfrak{f}}$, for all $w \in A^+$ and $1 \leq i \leq |w|$ we have:

$$\mathcal{U}^{w,i}(\text{Old}_x) = I_{\mathfrak{f}} T_{\mathfrak{f}}(w[1:i])(x).$$

We can finally show that the registers Sum_x store the information we wanted.

▷ **Claim 52.** Given $x \in \mathfrak{X}_{\mathfrak{f}}$, for all $w \in A^+$ and $0 \leq i \leq |w|$ we have:

$$\mathcal{U}^{w,i}(\text{Sum}_x) = \sum_{\substack{1 \leq j \leq i \\ j \in \text{Calls}(w, \mathfrak{f})}} I_{\mathfrak{f}} T_{\mathfrak{f}}(w[1:i] \uparrow j)(x).$$

Proof. We proceed by induction on $0 \leq i \leq |w|$. For $i = 0$ both terms equal 0. For the induction step with $i \geq 1$ let $\mathfrak{f} \in \mathfrak{F}$ and $x \in \mathfrak{X}_{\mathfrak{f}}$.

Suppose that $\lambda(\mu(w[1:i-1]), w[i], \mu(w[i+1:|w|])) = \mathfrak{f}$ (the case when they differ is similar and even easier), then:

$$\sum_{\substack{1 \leq j \leq i \\ j \in \text{Calls}(w, \mathfrak{f})}} I_{\mathfrak{f}} T_{\mathfrak{f}}(w[1:i] \uparrow j)(x) = I_{\mathfrak{f}} T_{\mathfrak{f}}(w[1:i] \uparrow i)(x) + \sum_{\substack{1 \leq j \leq i-1 \\ j \in \text{Calls}(w, \mathfrak{f})}} I_{\mathfrak{f}} T_{\mathfrak{f}}(w[1:i] \uparrow j)(x). \quad (1)$$

- Let $x \leftarrow \sum_{y \in \mathfrak{X}_i} \alpha_y y$ be the update performed by \mathcal{T}_i when reading a .
Then for $j \leq i - 1$, $I_i \mathcal{T}_i(w[1:i] \uparrow j)(x) = \sum_{y \in \mathfrak{X}_i} \alpha_y \times I_i \mathcal{T}_i(w[1:i-1] \uparrow j)(y)$.
- Let $x \leftarrow \sum_{y \in \mathfrak{X}_i} \beta_y y$ be the update performed by \mathcal{T}_i when reading \underline{a} .
Then $I_i \mathcal{T}_i(w[1:i] \uparrow i)(x) = \sum_{y \in \mathfrak{X}_i} \beta_y \times I_i \mathcal{T}_i(w[1:i-1])(y)$.

Hence we can rewrite Equation 1 using values in position $i-1$. By Claim 51 and the induction hypothesis, this sum coincides with the update in \mathcal{U} . \triangleleft

The fact that \mathcal{U} computes f follows from the definition of F and Claim 48.

Graph Characterization of the Universal Theory of Relations

Amina Doumane ✉

CNRS, ENS Lyon, France

Abstract

The equational theory of relations can be characterized using graphs and homomorphisms. This result, found independently by Freyd and Scedrov and by Andr eka and Bredikhin, shows that the equational theory of relations is decidable. In this paper, we extend this characterization to the whole universal first-order theory of relations. Using our characterization, we show that the positive universal fragment is also decidable.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases Relation algebra, Graph homomorphism, Equational theories, First-order logic

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.41

Acknowledgements I am deeply grateful to Denis Kuperberg and Damien Pous for their great help.

1 Introduction

Binary relations are a versatile mathematical object, used to model graphs, programs, databases, etc. It is then a natural task to understand the laws governing them. Since the seminal work of Tarski [14], this task has occupied researchers for several decades [15, 12, 3, 9, 11, 10, 2, 4, 13].

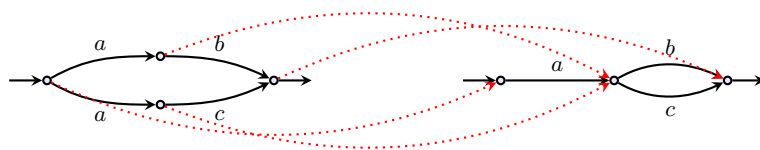
Relations usually come with a certain number of standard operations: union \cup , intersection \cap , composition \cdot , converse $^\circ$ etc. We are interested in containment between terms built with these operations with respect to their relational interpretations. When a containment between two terms t and u holds, we say that $t \geq u$ is a *valid inequation for relations* and write $\mathcal{R}el \models t \geq u$. For instance, an emblematic valid inequation is the following one:

$$(a \cdot b) \cap (a \cdot c) \geq a \cdot (b \cap c)$$

This law is valid because no matter how we interpret the letters a, b and c as relations, the relation denoted by the term $a \cdot (b \cap c)$ will be contained in the relation denoted by the term $(a \cdot b) \cap (a \cdot c)$. A very simple way to check that this inequation is valid relies on the following characterization ([1, Thm. 1], [7, p. 208]):

$$\mathcal{R}el \models t \geq u \quad \Leftrightarrow \quad \mathcal{G}(t) \triangleright \mathcal{G}(u) \quad (\star)$$

In this theorem, $\mathcal{G}(t)$ and $\mathcal{G}(u)$ are finite graphs associated to the terms t and u respectively, and \triangleright denotes the existence of a *graph homomorphism*. For example, the validity of the law above is witnessed by this homomorphism (in red) from the graph of $(a \cdot b) \cap (a \cdot c)$ to the graph of $a \cdot (b \cap c)$:



© Amina Doumane;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 41; pp. 41:1–41:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Using this characterization, we can show that relational validity of inequations is decidable.

As maybe noticed by the reader, inequations are implicitly universally quantified. They actually form a fragment of the more general *universal first-order* formulas. The latter comprises *universal positive formulas* which are basically disjunctions of inequations, and *Horn formulas* which are implications between inequations.

Universal first-order formulas have received a lot of attention in the model theory community. They enjoy for example the Łoś–Tarski theorem [8, Thm.5.4.4], which states that the set of universal first-order formulas is exactly the set of first-order formulas preserved under taking substructures.

In this paper, we give a graph characterization for those universal first-order formulas which are valid for relations, generalizing the characterization (\star). To this end, we proceed in three steps. First, we provide a characterization of relational validity for positive universal formulas. Based on this, we show that relational validity is decidable for this fragment. As a second step, we characterize relational validity for Horn formulas. Finally we combine the techniques used for both fragments to characterize validity for all universal first-order formulas. Before presenting our results, we start by recalling some background in Section 2.

2 Preliminaries

2.1 Universal theory of relations

We let $a, b \dots$ range over the letters of an alphabet A . *Terms* are generated by this syntax:

$$t, u ::= t \cdot u \mid t \cap u \mid t^\circ \mid 1 \mid \top \mid a \quad a \in A$$

We denote the set of terms by \mathcal{T} . We often write tu for $t \cdot u$, and assign priorities to symbols so that $ab \cap c$, $a \cap b^\circ$ and ab° parse respectively as $(a \cdot b) \cap c$, $a \cap (b^\circ)$ and $a \cdot (b^\circ)$.

First-order formulas are generated by the following syntax:

$$\varphi, \psi ::= t \geq u \mid \neg(t \geq u) \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists a. \varphi \mid \forall a. \varphi \quad t, u \in \mathcal{T}, a \in A.$$

Formulas of the form $t \geq u$ are called *inequations*. We extend the operation of negation \neg to all formulas in the standard way, for instance $\neg(\varphi \wedge \psi) = \neg\varphi \vee \neg\psi$. Implication $\varphi \Rightarrow \psi$ is a shortcut for $\neg\varphi \vee \psi$. Free and bound variables are defined as usual, and we call *sentence* a formula without free variables.

A *universal formula* is a formula from the syntax above which does not use existential quantification. A *generalized Horn formula* is a formula of the following form, where $\forall \vec{a}$ denotes a sequence of universal quantifications:

$$\forall \vec{a}. \bigwedge_{j \in J} (v_j \geq w_j) \Rightarrow \bigvee_{i \in I} (t_i \geq u_i)$$

We generally write it as follows, where \mathcal{H} is the set of inequations $\{v_j \geq w_j, j \in J\}$:

$$\forall \vec{a}. \mathcal{H} \Rightarrow \bigvee_{i \in I} (t_i \geq u_i)$$

We call \mathcal{H} its *hypothesis* and $\bigvee_{i \in I} (t_i \geq u_i)$ its *conclusion*. A *Horn formula* is a generalized Horn formula whose conclusion contains a single disjunct. We write it like this:

$$\forall \vec{a}. \mathcal{H} \Rightarrow t \geq u$$

A *Positive universal formula* is a generalized Horn formula whose set of hypothesis is empty. It looks like this:

$$\forall \vec{a}. \bigvee_{i \in I} (t_i \geq u_i)$$

A *universal inequation* is a positive universal formula with a single disjunct. We will sometimes call it simply inequation. It looks like this:

$$\forall \vec{a}. t \geq u$$

In the rest of the paper, we will be interested only on universal sentences, this is why we will omit the universal quantification in front of our formulas.

Note that every universal formula can be written as the conjunction of generalized Horn formulas. In the rest of the paper, we will mainly focus on the latter.

Let us define *relational validity* for generalized Horn sentences. An *interpretation* σ is a function $\sigma : A \rightarrow \mathcal{P}(B \times B)$ mapping letters into relations over a base set B . We can extend σ to all terms $\sigma : \mathcal{T} \rightarrow \mathcal{P}(B \times B)$, by interpreting the operations $\cdot, \cap, \circ, 1$ and \top on relations as follows:

$$\begin{aligned} R \cdot S &= \{(x, y) \mid \exists z. (x, z) \in R \text{ and } (z, y) \in S\} && \text{(Composition)} \\ R \cap S &= \{(x, y) \mid (x, y) \in R \text{ and } (x, y) \in S\} && \text{(Intersection)} \\ R^\circ &= \{(x, y) \mid (y, x) \in R\} && \text{(Converse)} \\ 1 &= \{(x, x) \mid x \in B\} && \text{(Identity)} \\ \top &= \{(x, y) \mid x, y \in B\} && \text{(Full relation)} \end{aligned}$$

Let σ be an interpretation as above. An inequation $t \geq u$ is *true under* σ , noted $\sigma \models t \geq u$, if $\sigma(t) \supseteq \sigma(u)$. A set of inequations \mathcal{H} are *true under* σ , noted $\sigma \models \mathcal{H}$, if this is the case for every inequation in \mathcal{H} . A generalized Horn sentence

$$\varphi := (\mathcal{H} \Rightarrow \bigvee_{i \in I} (t_i \geq u_i))$$

is *true under* σ , noted $\sigma \models \varphi$ if either $\sigma \not\models \mathcal{H}$ or there exists $i \in I$ such that $\sigma \models t_i \geq u_i$. We say that φ is *valid* for relations, noted $\mathcal{R}el \models \varphi$, if φ is true under all interpretations, using all possible base sets B .

Here are respectively a universal inequation (1), a positive universal sentence (2), and a Horn sentence (3), that are all valid for relations:

$$a(ba \cap 1)b \geq ab \cap 1 \tag{1}$$

$$(\top c \top \cap ab \cap ad \geq a(b \cap d)) \vee (d \geq ac) \tag{2}$$

$$ef^\circ \geq \top \Rightarrow (ae \cap cf)(e^\circ b \cap f^\circ d) \geq ab \cap cd \tag{3}$$

We will see in the upcoming sections how to check their validity.

2.2 Graph characterization of the inequational theory of relations

Let A be an alphabet. A *2-pointed labeled graph* is a structure (V, E, ι, o) where V is a set of vertices, $E \subseteq V \times A \times V$ is a set of edges and ι and o are two distinguished vertices called the *input* and *output*. We simply call them *graphs* in the sequel; we depict them as expected, with unlabeled ingoing and outgoing arrows to denote the input and the output, respectively. We denote by $\mathcal{G}r$ the set of finite graphs. If G is a graph and x, y two of its vertices, we denote by (x, G, y) the graph obtained from G by forgetting the original input and output of G , and considering x and y as the new input and output respectively.

41:4 Graph Characterization of the Universal Theory of Relations

We define the following operations of graphs.

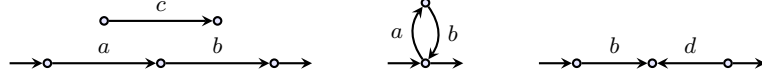
$$G \cap H = \begin{array}{c} \circ \\ \swarrow G \\ \circ \quad \circ \\ \searrow H \\ \circ \end{array} \quad G \cdot H = \circ \rightarrow G \rightarrow \circ \rightarrow H \rightarrow \circ \quad G^\circ = \circ \leftarrow G \rightarrow \circ$$

We associate to every term $t \in \mathcal{T}$ a graph $\mathcal{G}(t)$ called the *graph of t* , by letting

$$\mathcal{G}(a) = \circ \xrightarrow{a} \circ \quad \mathcal{G}(1) = \circ \rightarrow \circ \quad \mathcal{G}(\top) = \circ \quad \circ \rightarrow$$

and by interpreting the operations \cdot, \cap and $^\circ$ on graphs as above.

► **Example 1.** The graphs $\mathcal{G}(\top \circ \top \cap ab)$, $\mathcal{G}(ab \cap 1)$ and $\mathcal{G}(bd^\circ)$ are respectively the following:



Graph homomorphisms play a central role in the paper, they are defined as follows:

► **Definition 2** (Graph homomorphism). *Given two graphs $G = \langle V, E, \iota, o \rangle$ and $G' = \langle V', E', \iota', o' \rangle$, a (graph) homomorphism $h : G \rightarrow H$ is a mapping from $V \rightarrow V'$ that preserves labeled edges, ie. if $(x, a, y) \in E$ then $(h(x), a, h(y)) \in E'$, and preserves input and output, ie. $h(\iota) = \iota'$ and $h(o) = o'$.*

The image of G by h , denoted $h(G)$, is the graph $\langle h(V), E'', \iota', o' \rangle$ where

$$E'' = \{(h(x), a, h(y)) \mid (x, a, y) \in E\}.$$

We write $G \triangleright H$ if there exists a graph homomorphism from G to H , and $G \hookrightarrow H$ if there exists an injective graph homomorphism from G to H . In the later case, we usually consider G as an actual subgraph of H .

Our starting point was this characterization of the inequational theory of relations:

► **Theorem 3** ([1, Thm. 1], [7, p. 208]). *For all terms u, v ,*

$$\mathcal{R}el \models u \geq v \quad \text{iff} \quad \mathcal{G}(u) \triangleright \mathcal{G}(v)$$

2.3 Graphs and interpretations

We state below the main lemma (Lemma 6) that was used to prove Theorem 3, which will be useful for us too. But first, let us explicit a link between graphs and interpretations.

► **Definition 4** (Graphs and interpretations). *Let $\sigma : A \rightarrow \mathcal{P}(B \times B)$ be an interpretation. The graph associated to σ , $\mathcal{G}(\sigma)$, is the graph whose set of vertices is B and*

$$(x, a, y) \text{ is an edge of } \mathcal{G}(\sigma) \quad \text{iff} \quad (x, y) \in \sigma(a).$$

Conversely if $G = (V, E)$ is a graph, the interpretation associated to G , $\mathcal{I}(G)$, is the function

$$\begin{aligned} A &\rightarrow \mathcal{P}(V \times V) \\ a &\mapsto \{(x, y) \mid (x, a, y) \in E\} \end{aligned}$$

In the above definition, graphs are considered without distinguished input and output.

► **Remark 5.** The functions \mathcal{G} and \mathcal{I} are inverses of each other: $\mathcal{I} \circ \mathcal{G}$ and $\mathcal{G} \circ \mathcal{I}$ are the identity function on interpretations and graphs respectively.

Recall that (x, G, y) is the graph G where x and y are chosen to be the input and output.

► **Lemma 6** ([1, Lemma 3]). *Let t be a term, $\sigma : A \rightarrow \mathcal{P}(B \times B)$ be an interpretation and $x, y \in B$. We have that:*

$$\sigma(t) \ni (x, y) \quad \text{iff} \quad \mathcal{G}(t) \triangleright (x, \mathcal{G}(\sigma), y)$$

3 Characterizing the positive universal theory of relations

Given two graphs G and H , we define $G \oplus H$ as the disjoint union of G and H , whose input and output are those of G . Note that \oplus is associative, but not commutative. However, note that the following holds:

$$G \oplus H \oplus K = G \oplus K \oplus H$$

$$G \triangleright H \oplus H \oplus K \Leftrightarrow G \triangleright H \oplus K$$

Now we can state our first characterization theorem:

► **Theorem 7.** For all terms t_i, u_i where $i \in [1, n]$, the following holds

$$\mathcal{R}el \models \bigvee_{i \in [1, n]} (t_i \geq u_i) \quad \text{iff} \quad \bigvee_{i \in [1, n]} (\mathcal{G}(t_i) \triangleright \mathcal{G}(u_i) \oplus G)$$

where $G = \mathcal{G}(u_1) \oplus \dots \oplus \mathcal{G}(u_n)$.

Using the remark above, the case of two disjuncts can be formulated as follows

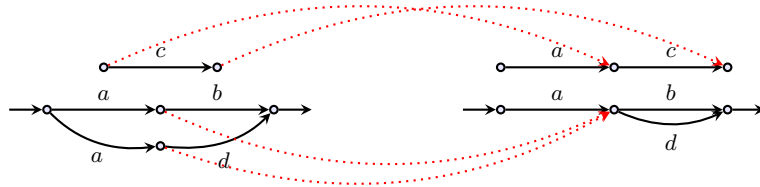
$$\mathcal{R}el \models (t_0 \geq u_0) \vee (t_1 \geq u_1) \quad \text{iff} \quad \mathcal{G}(t_0) \triangleright \mathcal{G}(u_0) \oplus \mathcal{G}(u_1) \quad \text{or} \quad \mathcal{G}(t_1) \triangleright \mathcal{G}(u_1) \oplus \mathcal{G}(u_0)$$

Before proving Theorem 7, let us see an example of its application.

► **Example 8.** The validity of the following positive universal sentence

$$(\top c \top \cap ab \cap ad \geq a(b \cap d)) \vee (d \geq ac) \quad (2)$$

is witnessed by this homomorphism depicted below:



► **Remark 9.** Surprisingly, this characterization tells us that only one left-hand-side (lhs) of the disjuncts of a positive universal sentence plays a role in its validity. For instance, in the sentence (2) above, we can replace the lhs of the second inequation, d , by any term without affecting the validity.

Proof. We show here the case of binary disjunctions to lighten notations. The general case works exactly in the same way.

(\Rightarrow) Suppose that $\mathcal{R}el \models (t_0 \geq u_0) \vee (t_1 \geq u_1)$, let us show that either

$$\mathcal{G}(t_0) \triangleright \mathcal{G}(u_0) \oplus \mathcal{G}(u_1) \quad \text{or} \quad \mathcal{G}(t_1) \triangleright \mathcal{G}(u_1) \oplus \mathcal{G}(u_0)$$

Let G be the graph (without specified input and output) which is the disjoint union of $\mathcal{G}(u_0)$ and $\mathcal{G}(u_1)$, and let σ be the interpretation associated to G . We denote by G_0 the graph $\mathcal{G}(u_0) \oplus \mathcal{G}(u_1)$ and by G_1 the graph $\mathcal{G}(u_1) \oplus \mathcal{G}(u_0)$. To conclude the proof of this direction, we show that, for $i = 0, 1$:

$$\sigma(t_i) \supseteq \sigma(u_i) \quad \Rightarrow \quad \mathcal{G}(t_i) \triangleright G_i$$

41:6 Graph Characterization of the Universal Theory of Relations

Suppose that $\sigma(t_0) \supseteq \sigma(u_0)$, the other case is treated symmetrically. Let ι and o be respectively the vertices corresponding to the input and output of $\mathcal{G}(u_0)$ in G . We have that $\mathcal{G}(u_0) \triangleright (\iota, G, o)$, then by Lemma 6, $\sigma(u_0) \ni (\iota, o)$. Thus, $\sigma(t_0) \ni (\iota, o)$ and again by Lemma 6, $\mathcal{G}(t_0) \triangleright (\iota, G, o)$. But (ι, G, o) is G_0 and this remark concludes the proof.

(\Leftarrow) Suppose that $G(t_0) \triangleright \mathcal{G}(u_0) \oplus \mathcal{G}(u_1)$ and let us show that:

$$\mathcal{R}el \models (t_0 \geq u_0) \vee (t_1 \geq u_1)$$

The other case is treated symmetrically. Let $\sigma : A \rightarrow \mathcal{P}(B \times B)$ be an interpretation, and let G be its graph. We distinguish two cases. We have either:

$$\forall x, y \in B, \quad \mathcal{G}(u_1) \not\triangleright (x, G, y)$$

In this case, by Lemma 6, there is no pair (x, y) such that $(x, y) \in \sigma(u_1)$, hence $\sigma(t_1) \supseteq \sigma(u_1)$ is vacuously true.

Suppose now that there is x_1 and y_1 in B such that $\mathcal{G}(u_1) \triangleright (x_1, G, y_1)$, let h_1 be such homomorphism. Notice the following:

$$\forall x, y \in B, \quad \mathcal{G}(u_0) \triangleright (x, G, y) \Rightarrow \mathcal{G}(u_0) \oplus \mathcal{G}(u_1) \triangleright (x, G, y) \quad (\dagger)$$

Indeed, if h_0 is a homomorphism from $\mathcal{G}(u_0)$ to (x, G, y) , then we can combine it with h_1 to get a homomorphism from $\mathcal{G}(u_0) \oplus \mathcal{G}(u_1)$ to (x, G, y) .

Let us show that $\sigma(t_0) \supseteq \sigma(u_0)$. If $\sigma(u_0) \ni (x, y)$, then by Lemma 6, we have that $\mathcal{G}(u_0) \triangleright (x, G, y)$. Using the remark (\dagger), we get that $\mathcal{G}(u_0) \oplus \mathcal{G}(u_1) \triangleright (x, G, y)$. By our hypothesis, we know that $\mathcal{G}(t_0) \triangleright \mathcal{G}(u_0) \oplus \mathcal{G}(u_1)$, thus $\mathcal{G}(t_0) \triangleright (x, G, y)$. We conclude that $\sigma(t_0) \ni (x, y)$, and this ends the proof of our first characterization theorem. \blacktriangleleft

Testing the existence of a homomorphism between finite graphs is decidable. Hence, we get as a corollary of Theorem 7 that:

► **Theorem 10.** *The positive universal theory of relations is decidable.*

4 Characterizing the Horn theory of relations

To give a characterization of the Horn theory of relations, we need to generalize the homomorphism relation between graphs to take into account some set of hypothesis.

A *context* is a graph with a distinguished edge labeled by a special letter \bullet , called its *hole*. If G is a graph and C a context, then $C[G]$ is the graph obtained by “plugging G in the hole” of C , that is, $C[G]$ is the graph obtained as the disjoint union of G and C , where we identify the input (resp. output) of G with the input (resp. output) of the edge labeled by \bullet in C , and we remove the edge of C labeled \bullet .

► **Definition 11** (The relation $\triangleright_{\mathcal{H}}$). *Let \mathcal{H} be a set of inequations. We define the relation $\triangleright_{\mathcal{H}}$ on graphs as follows. We set $G \triangleright_{\mathcal{H}} H$ if and only if there is a context C and an inequation $(t \geq u) \in \mathcal{H}$ such that*

$$G = C[\mathcal{G}(t)] \quad \text{and} \quad H = C[\mathcal{G}(u)]$$

We define $\triangleright_{\mathcal{H}}$ as the transitive closure of $\triangleright \cup \triangleright_{\mathcal{H}}$.

In the definition above, the graphs G , H and C are not necessarily the graphs of some terms.

We can state now the main theorem of this section:

► **Theorem 12.** For all terms t, u and set of inequations \mathcal{H} , we have:

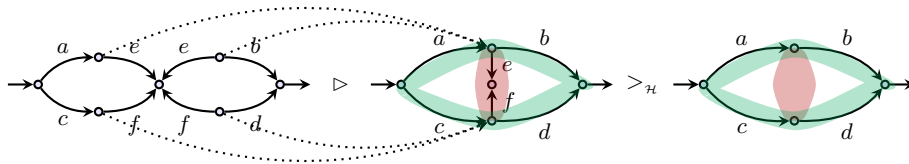
$$\mathcal{R}el \models (\mathcal{H} \Rightarrow t \geq u) \quad \text{iff} \quad \mathcal{G}(t) \triangleright_{\mathcal{H}} \mathcal{G}(u)$$

Hence, in order to show that a Horn sentence $(\mathcal{H} \Rightarrow t \geq u)$ is valid, we need to find a sequence of graphs G_0, \dots, G_n such that $G_0 = \mathcal{G}(t)$, $G_n = \mathcal{G}(u)$ and for every $i \in [0, n - 1]$ the graphs G_i and G_{i+1} are either related by homomorphism or by the relation $\triangleright_{\mathcal{H}}$. We say that this sequence *witnesses* the validity of this Horn sentence.

► **Example 13.** The validity of the following Horn sentence:

$$ef^\circ \geq \top \Rightarrow (ae \cap cf)(e^\circ b \cap f^\circ d) \geq ab \cap cd \quad (3)$$

is witnessed by the following sequence:



We start by applying a homomorphism represented by the dotted lines, then we factorize the obtained graph into a context (in green) and an inner graph (in red) which is the graph of ef° , the lhs of the hypothesis $ef^\circ \geq \top$. We replace it by the graph of the rhs \top , which is the empty graph. Doing so, we get the graph of $ab \cap cd$

Notice that the intermediary graph is *not* the graph of a term.

► **Remark 14.** One may wonder whether Theorem 12 leads to a decidability result for the Horn theory of relations. Actually, the latter is undecidable, as it subsumes the word problem for monoids [6, Thm.4.5].

The next two subsections are dedicated to the proof of Theorem 12.

4.1 From $\triangleright_{\mathcal{H}}$ to validity

In this section we prove the right-to-left implication of Theorem 12. But first, let us show the following lemma, which says that $\triangleright_{\mathcal{H}}$ collapses to \triangleright if the target graph is the graph of an interpretation making \mathcal{H} true.

► **Lemma 15.** Let \mathcal{H} be a set of inequations and σ an interpretation. If the inequations \mathcal{H} are true under σ , then for every graph G :

$$G \triangleright_{\mathcal{H}} (x, G(\sigma), y) \quad \text{iff} \quad G \triangleright (x, G(\sigma), y)$$

Proof. The right-to-left direction is trivial. We prove the other direction by induction on the length of a sequence witnessing that $G \triangleright_{\mathcal{H}} (x, G(\sigma), y)$. The most interesting base case is when, for some graph H :

$$G \triangleright_{\mathcal{H}} H \triangleright (x, G(\sigma), y) \quad (\text{BC})$$

The other two base cases are: $G \triangleright (x, G(\sigma), y)$, which is trivial, and $G \triangleright_{\mathcal{H}} (x, G(\sigma), y)$, which can be seen as a particular case of the interesting base case, by taking H to be $(x, G(\sigma), y)$. The inductive step is easy, as the composition of two homomorphisms is a homomorphism. Now, let us prove the interesting base case. Suppose that there is a graph H satisfying (BC), and let us find a homomorphism from G to $(x, G(\sigma), y)$.

41:8 Graph Characterization of the Universal Theory of Relations

Since $G >_{\mathcal{H}} H$, there is an inequation $(t \geq u) \in \mathcal{H}$ and a context C such that $G = C[\mathcal{G}(t)]$ and $H = C[\mathcal{G}(u)]$. We have also that $H \triangleright (x, G(\sigma), y)$, so let h be a homomorphism:

$$h : C[\mathcal{G}(u)] \rightarrow (x, G(\sigma), y)$$

Let x' and y' be respectively the image of the input and the output of $\mathcal{G}(u)$ by h . By considering the restriction of h to $\mathcal{G}(u)$, we have that $\mathcal{G}(u) \triangleright (x', G(\sigma), y')$. Hence, by Lemma 6, we have that $(x', y') \in \sigma(u)$. As \mathcal{H} is true under σ , we have also that $(x', y') \in \sigma(t)$, and again by Lemma 6, $\mathcal{G}(t) \triangleright (x', G(\sigma), y')$. Let us denote by k a homomorphism:

$$k : \mathcal{G}(t) \rightarrow (x', G(\sigma), y')$$

With these ingredients, we construct a homomorphism f from $G = C[\mathcal{G}(t)]$ to $(x, G(\sigma), y)$ as follows: the restriction of f to C is h and the restriction of f to $\mathcal{G}(t)$ is k . It is easy to check that f is indeed an homomorphism, and this ends the proof. \blacktriangleleft

We can now prove the right-to-left direction of Theorem 12.

Proof of Theorem 12 (\Leftarrow). Suppose that $\mathcal{G}(t) \triangleright_{\mathcal{H}} \mathcal{G}(u)$. Let σ be an interpretation satisfying \mathcal{H} and suppose that $(x, y) \in \sigma(u)$.

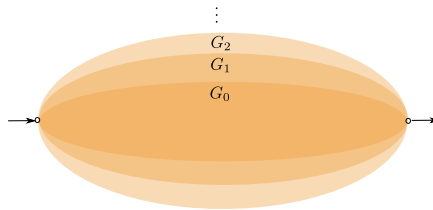
$\sigma(u) \ni (x, y)$	$\Rightarrow \mathcal{G}(u) \triangleright (x, G(\sigma), y)$	Lem. 6
	$\Rightarrow \mathcal{G}(t) \triangleright_{\mathcal{H}} (x, G(\sigma), y)$	By hypothesis
	$\Rightarrow \mathcal{G}(t) \triangleright (x, G(\sigma), y)$	Lem. 15
	$\Rightarrow \sigma(t) \ni (x, y)$	Lem. 6 \blacktriangleleft

4.2 From validity to $\triangleright_{\mathcal{H}}$

The main ingredient to prove the left-to-right direction of Theorem 12 is to construct, given a set of hypothesis \mathcal{H} , an interpretation making them true. For that we start from an arbitrary graph and “saturate” it by the hypothesis \mathcal{H} , then we iterate this construction ω -times and take the *limit graph*. The desired interpretation will be the interpretation associated to this graph. In the sequel, we define the notions of graph limit and saturation, then we proceed to the proof of our theorem.

4.2.1 Limit of a sequence of graphs

When we consider an increasing sequence of graphs $(G_i)_{i \in \omega}$, that is, $G_i \hookrightarrow G_{i+1}$ for every $i \in \omega$, the notion of limit is clear: it is just the union of the graphs G_i , its input and output being respectively the common input and output of the graphs G_i ; we denote it by $\lim_{i \in \omega} G_i$. We denote by $\theta_i : G_i \rightarrow \lim_{i \in \omega} G_i$ the natural injection of G_i into the limit graph, we call it the *limit injection for G_i* . Here is an illustration of this construction:



In the following, we extend this notion of limit to the case where the graphs G_i and G_{i+1} are related by an arbitrary homomorphism, not necessarily an injective one. Let us start with an observation.

Let $G_0 \xrightarrow{h_0} G_1 \xrightarrow{h_1} G_2 \dots$ be a sequence of *finite* graphs related by homomorphism. Let $(H_i)_{i \in \omega}$ be the successive images of G_0 by these homomorphisms, that is:

$$H_0 = G_0, \quad \text{and} \quad H_{i+1} = h_i(H_i) \quad \text{for } i \geq 0.$$

At some point, the image of G_0 will stabilize, in other words there is an index s such that, for all $i > s$ the function $k_i : H_i \rightarrow H_{i+1}$, the restriction of h_i to H_i is a bijection. We call *stabilization index* of G_0 the least index s satisfying this property, we denote it by s_0 . We call *stable image* of G_0 the graph H_{s_0} and we denote it by $S(G_0)$.

We define in the same way the *stabilization index* of G_i , and denote it s_i : it is the least index starting from which the homomorphisms h_j for $j > s_i$ do not merge nodes coming from G_i . We define similarly the stable image of G_i and denote it by $S(G_i)$.

Note that if $i \leq j$ then $s_i \leq s_j$ and $S(G_i) \hookrightarrow S(G_j)$. By considering the sequence of the stable images of the graphs G_i , we can now define the limit of this sequence:

► **Definition 16** (Limit of a sequence of graphs). *Let $(G_i)_{i \in \omega}$ be a sequence of finite graphs such that there is a homomorphism $h_i : G_i \rightarrow G_{i+1}$ for every $i \in \omega$. As the sequence of stable images $(S(G_i))_{i \in \omega}$ is increasing, its limit $\lim_{i \in \omega} S(G_i)$ is well defined. For every $i \in \omega$, let θ_i be the limit injection $\theta_i : S(G_i) \rightarrow \lim_{i \in \omega} S(G_i)$.*

We define the limit of the sequence $(G_i)_{i \in \omega}$ as follows:

$$\lim_{i \in \omega} G_i = \lim_{i \in \omega} S(G_i)$$

For every $i < j \in \omega$, we denote by $h_{[i,j]}$ the homomorphism $h_{[i,j]} : G_i \rightarrow G_j$ obtained as the composition $h_{j-1} \circ \dots \circ h_i$. We denote by $\pi_i : G_i \rightarrow \lim_{i \in \omega} G_i$ the homomorphism $\theta_{s_i} \circ h_{[i,s_i]}$. We call π_i the limit homomorphism for G_i .

► **Example 17.** Consider the sequence of terms $(t_i)_{i \in \omega}$ defined by:

$$t_i = \left(\bigcap_{k=0}^i a_k \cdot \bigcap_{k=0}^i b_k \right) \cap (a_{i+1} \cdot b_{i+1}).$$

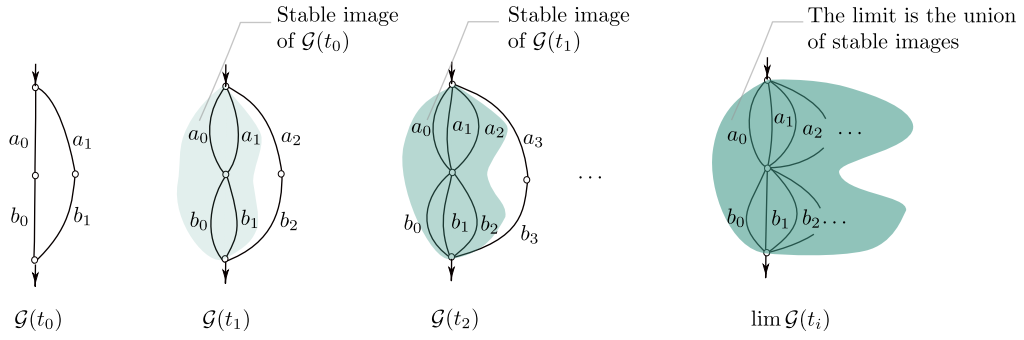
There is a (unique) homomorphism $h_i : \mathcal{G}(t_i) \rightarrow \mathcal{G}(t_{i+1})$. The limit of the sequence of graphs $(\mathcal{G}(t_i))_{i \in \omega}$ related by the homomorphisms $(h_i)_{i \in \omega}$, converges to the graph of this “term”¹:

$$\bigcap_{k=0}^{\infty} a_k \cdot \bigcap_{k=0}^{\infty} b_k$$

Here is an illustration of this example:

¹ This is not really a term since it contains infinite intersections, but it is clear how to define the graphs of such generalized terms.

41:10 Graph Characterization of the Universal Theory of Relations



Note that the limit does not depend only on the sequence of graphs, but also on the homomorphisms relating them. Consider for instance the following sequence of terms.

$$u_i = \left(\bigcap_{k=0}^i a_k \cdot \bigcap_{k=0}^i b_k \right) \cap \bigcap_{k=0}^{i+1} (a_k \cdot b_k).$$

If we consider the injections $\iota_i : \mathcal{G}(u_i) \rightarrow \mathcal{G}(u_{i+1})$, then the sequence $(\mathcal{G}(u_i))_{i \in \omega}$ related by the homomorphisms $(\iota_i)_{i \in \omega}$ converges to the graph of this “term”:

$$\left(\bigcap_{k=0}^{\infty} a_k \cdot \bigcap_{k=0}^{\infty} b_k \right) \cap \bigcap_{k=0}^{\infty} (a_k \cdot b_k)$$

But if we consider the homomorphisms $k_i : \mathcal{G}(u_i) \rightarrow \mathcal{G}(u_{i+1})$ which merges all the inner nodes² of $\mathcal{G}(u_i)$, we obtain as limit the graph of this “term”:

$$\bigcap_{k=0}^{\infty} a_k \cdot \bigcap_{k=0}^{\infty} b_k$$

► **Remark 18.** This notion of limit is a well known concept of category theory. Since the category of graphs and graph homomorphisms is cocomplete, every infinite sequence of homomorphisms has a colimit, unique up to isomorphism. We made the choice to give an explicit definition for the readers which are, as the author, not familiar with category theory.

Here are some properties satisfied by the limit of a sequence of graphs.

► **Proposition 19.** Let $(G_i)_{i \in \omega}$ be a sequence of graphs, and $h_i : G_i \rightarrow G_{i+1}$. Let G_ω be their graph limit, π_i be the limit homomorphism for G_i and H be a finite graph.

1. For every $i \in \omega$, if $H \triangleright (x, G_i, y)$ then $H \triangleright (\pi_i(x), G_\omega, \pi_i(y))$.
2. Conversely, if $H \triangleright (x, G_\omega, y)$ then $H \triangleright (x', G_i, y')$ for some i, x', y' satisfying $\pi_i(x') = x$ and $\pi_i(y') = y$.
3. In particular, we have that: $H \triangleright G_\omega \Leftrightarrow \exists i \in \omega, H \triangleright G_i$.

Proof. Property (1) is trivial. Indeed, if $h : H \rightarrow (x, G_i, y)$ is a homomorphism then $h_i \circ h : H \rightarrow (\pi_i(x), G_\omega, \pi_i(y))$ is also a homomorphism.

Suppose that $H \triangleright (x, G_\omega, y)$. Since H is finite, there is $i \in \omega$ such that $H \triangleright (x', S(G_i), y')$ where $\pi_i(x') = x$ and $\pi_i(y') = y$. But $S(G_i)$ is a subgraph of some G_j , where $j \in \omega$. Hence $H \triangleright (x', G_j, y')$. ◀

² That is, nodes different from the input and the output.

4.2.2 Saturation by hypothesis

Let $G, H \in \mathcal{Gr}$ and let x, y be two vertices of G . We denote by $G[H/xy]$ the graph obtained from G by merging the input of H with x and its output with y . The input and output of $G[H/xy]$ are those of G .

► **Remark 20.** Note that if the input and output of H are equal, then the operation $G[H/xy]$ merges the nodes x, y . Note also that $G \triangleright G[H/xy]$, but this homomorphism is not necessarily injective because of the possible merge of x and y .

► **Definition 21 (Saturation).** Let \mathcal{H} be a finite set of inequations, $G \in \mathcal{Gr}$ and V its set of vertices. Let $T \subseteq V \times V \times \mathcal{Gr}$ be the set of triplets satisfying:

$$(x, y, H) \in T \quad \text{iff} \quad \exists (t \geq u) \in \mathcal{H}, \quad \mathcal{G}(u) \triangleright (x, G, y) \quad \text{and} \quad H = \mathcal{G}(t)$$

Let $(x_i, y_i, H_i)_{i \leq n}$ be an enumeration of T . The saturation of G by \mathcal{H} is the graph denoted $Sat_{\mathcal{H}}(G)$ and defined as:

$$Sat_{\mathcal{H}}(G) = G[H_0/x_0y_0] \dots [H_n/x_ny_n]$$

In words, a triplet (x, y, H) is in T means that in the graph G , we “identified” the graph $\mathcal{G}(u)$, the rhs of a hypothesis in \mathcal{H} , between the nodes x and y . The graph H is $\mathcal{G}(t)$, the graph of the lhs of this hypothesis. To make G “agree” with hypothesis \mathcal{H} , we need to plug H between x and y . When we do that for all the triplets in T , we obtain the saturation of G by \mathcal{H} . Now, let us make some properties of saturation explicit.

► **Proposition 22.** Let G be a graph and \mathcal{H} a set of inequations.

1. For every inequation $(t \geq u) \in \mathcal{H}$, we have:

$$\mathcal{G}(u) \triangleright (x, G, y) \Rightarrow \mathcal{G}(t) \triangleright (x, Sat_{\mathcal{H}}(G), y)$$

2. $G \triangleright Sat_{\mathcal{H}}(G)$.

3. $Sat_{\mathcal{H}}(G) \triangleright_{\mathcal{H}} G$.

Proof. To prove (1), suppose that $(t \geq u) \in \mathcal{H}$ and $\mathcal{G}(u) \triangleright (x, G, y)$. This means that the triplet $(x, y, \mathcal{G}(t))$ is in the set T of Definition 21. Hence $Sat_{\mathcal{H}}(G)$ is of the form $K[\mathcal{G}(t)/xy]$. It is clear then that $\mathcal{G}(t) \triangleright (x, Sat_{\mathcal{H}}(G), y)$.

Property (2) is a consequence of Remark 20 above. For property (3), we will show that if $(x, y, H) \in T$, where T is as in definition 21, then $G[H/xy] \triangleright_{\mathcal{H}} G$. The result will be an iteration of this argument for all elements of T . By definition of T , there is a hypothesis $(t \geq u)$ such that $\mathcal{G}(u) \triangleright (x, G, y)$ and $H = \mathcal{G}(t)$. Let us denote by k a homomorphism from $\mathcal{G}(u)$ to (x, G, y) . Let C be the context obtained from G by adding an edge labeled \bullet between x and y . We have that:

$$G[H/xy] = C[\mathcal{G}(t)] \triangleright_{\mathcal{H}} C[\mathcal{G}(u)] \triangleright G$$

Indeed, the equality and inequation $\triangleright_{\mathcal{H}}$ are trivially true. To justify the \triangleright inequation, we define a homomorphism from $C[\mathcal{G}(u)]$ to G as follows: its restriction to $\mathcal{G}(u)$ is the homomorphism k , and its restriction to C is the identity. ◀

Now, we can define the ω -saturation of a graph by a set of hypothesis.

41:12 Graph Characterization of the Universal Theory of Relations

► **Definition 23** (ω -saturation). If G is a graph and \mathcal{H} a set of inequations, we define the sequence $(Sat_{\mathcal{H}}^i(G))_{i \in \omega}$ as the successive iterations of G by saturation by the hypothesis \mathcal{H} :

$$Sat_{\mathcal{H}}^0(G) = G, \quad Sat_{\mathcal{H}}^{i+1}(G) = Sat(Sat_{\mathcal{H}}^i(G)) \quad (i \in \omega).$$

By Proposition 22 (2), we have that $Sat_{\mathcal{H}}^i(G) \triangleright Sat_{\mathcal{H}}^{i+1}(G)$. The limit is then well defined by Definition 16. We define the ω -saturation of G by \mathcal{H} as the graph:

$$Sat_{\mathcal{H}}^{\omega}(G) = \lim_{i \in \omega} Sat_{\mathcal{H}}^i(G).$$

The ω -saturation satisfies the following property. It says that given a set of inequations \mathcal{H} , if we start from an arbitrary graph G (so in general, the inequations of \mathcal{H} are not true under the interpretation associated to G) and we ω -saturate it by \mathcal{H} , then the inequations from \mathcal{H} are true under the interpretation associated to the obtained graph.

► **Proposition 24.** Let \mathcal{H} be a set of inequations, G be a graph, and σ the interpretation associated to $Sat_{\mathcal{H}}^{\omega}(G)$. The inequations from \mathcal{H} are true under σ .

Proof. We denote by G^{ω} the graph $Sat_{\mathcal{H}}^{\omega}(G)$, by G^i the graph $Sat_{\mathcal{H}}^i(G)$ for every $i \in \omega$ and by σ the interpretation associated to G^{ω} . By Remark 5, the graph associated to σ is G^{ω} . Let $\pi_i : G^i \rightarrow G^{\omega}$ be the limit homomorphism for G_i .

Let $(t \geq u) \in \mathcal{H}$, let us show that $\sigma(t) \supseteq \sigma(u)$. Suppose that $\sigma(u) \ni (x, y)$.

$$\begin{aligned} \sigma(u) \ni (x, y) &\implies \mathcal{G}(u) \triangleright (x, G^{\omega}, y) && \text{Lem. 6} \\ &\xrightarrow{\exists i, x', y'} \mathcal{G}(u) \triangleright (x', G^i, y'), \quad x = \pi_i(x') \text{ and } y = \pi_i(y') && \text{Prop. 19 (2)} \\ &\implies \mathcal{G}(t) \triangleright (x', G^{i+1}, y') && \text{Prop. 22 (1)} \\ &\implies \mathcal{G}(t) \triangleright (x, G^{\omega}, y) && \text{Prop. 19 (1)} \\ &\implies \sigma(t) \ni (x, y) && \text{Lem. 6} \end{aligned}$$

And this concludes the proof. ◀

We can go back to the proof of Theorem 12.

Proof of Theorem 12 (\Leftarrow). Suppose that $\mathcal{R}el \models \mathcal{H} \Rightarrow t \geq u$ and let us show that $\mathcal{G}(t) \triangleright_{\mathcal{H}} \mathcal{G}(u)$. We denote by $\mathcal{G}(u)^{\omega}$ the graph $Sat_{\mathcal{H}}^{\omega}(\mathcal{G}(u))$, by $\mathcal{G}(u)^i$ the graph $Sat_{\mathcal{H}}^i(\mathcal{G}(u))$ for every $i \in \omega$, and by σ be the interpretation associated to $\mathcal{G}(u)^{\omega}$. By Proposition 24, the inequations \mathcal{H} are true under σ . Note that the graph associated to σ is $\mathcal{G}(u)^{\omega}$ and that the input and output of $\mathcal{G}(u)^{\omega}$ are those of $\mathcal{G}(u)$, let us denote them by ι and o respectively.

By Proposition 19 (1), we have $\mathcal{G}(u) \triangleright \mathcal{G}(u)^{\omega}$. It follows that:

$$\begin{aligned} \mathcal{G}(u) \triangleright \mathcal{G}(u)^{\omega} &\Rightarrow \sigma(u) \ni (\iota, o) && \text{Lem. 6} \\ &\Rightarrow \sigma(t) \ni (\iota, o) && \text{By hypothesis} \\ &\Rightarrow \mathcal{G}(t) \triangleright \mathcal{G}(u)^{\omega} && \text{Lem. 6} \\ &\Rightarrow \mathcal{G}(t) \triangleright \mathcal{G}(u)^i \text{ for some } i \in \omega && \text{Prop. 19 (3)} \\ &\Rightarrow \mathcal{G}(t) \triangleright_{\mathcal{H}} \mathcal{G}(u) && \text{Prop. 22 (3)} \end{aligned}$$

This ends the proof of Theorem 12. ◀

4.3 Characterizing the universal theory of relations

We characterize now the validity of the generalized Horn sentences. The proof is a mix of the techniques used to prove Theorems 7 and 12.

► **Theorem 25.** *For all terms t_i, u_i where $i \in [1, n]$, and set of inequations \mathcal{H} , the following holds:*

$$\mathcal{R}el \models \mathcal{H} \Rightarrow \bigvee_{i \in [1, n]} (t_i \geq u_i) \quad \text{iff} \quad \bigvee_{i \in [1, n]} (\mathcal{G}(t_i) \triangleright_{\mathcal{H}} \mathcal{G}(u_i) \oplus G)$$

where $G = \mathcal{G}(u_1) \oplus \dots \oplus \mathcal{G}(u_n)$.

Proof. As for Theorem 7, we prove this result in the case of binary disjunctions, that is:

$$\mathcal{R}el \models \mathcal{H} \Rightarrow (t_0 \geq u_0) \vee (t_1 \geq u_1) \quad \text{iff} \quad G(t_0) \triangleright_{\mathcal{H}} \mathcal{G}(u_0) \oplus G(u_1) \text{ or } G(t_1) \triangleright_{\mathcal{H}} \mathcal{G}(u_1) \oplus G(u_0)$$

(\Rightarrow) Suppose that $\mathcal{R}el \models \mathcal{H} \Rightarrow (t_0 \geq u_0) \vee (t_1 \geq u_1)$, let us show that either

$$G(t_0) \triangleright_{\mathcal{H}} \mathcal{G}(u_0) \oplus G(u_1) \quad \text{or} \quad G(t_1) \triangleright_{\mathcal{H}} \mathcal{G}(u_1) \oplus G(u_0)$$

We set $G = \mathcal{G}(u_0) \oplus \mathcal{G}(u_1)$, and let G^ω denote the graph $Sat_{\mathcal{H}}^\omega(G)$, G^i denote the graph $Sat_{\mathcal{H}}^i(G)$ for every $i \in \omega$, and let σ be the interpretation associated to G^ω . By Proposition 24, the inequations \mathcal{H} are true under σ . Hence, we have either $\sigma(t_0) \supseteq \sigma(u_0)$ or $\sigma(t_1) \supseteq \sigma(u_1)$. Let us study the former case, the latter being symmetric.

Suppose that $\sigma(t_0) \supseteq \sigma(u_0)$. Let ι and o be respectively the input and output of G^ω . Notice that $\mathcal{G}(u_0) \triangleright (\iota, G^\omega, o)$, it follows that:

$$\begin{aligned} \mathcal{G}(u_0) \triangleright (\iota, G^\omega, o) &\Rightarrow \sigma(u_0) \ni (\iota, o) && \text{Lem. 6} \\ &\Rightarrow \sigma(t_0) \ni (\iota, o) && \text{By hypothesis} \\ &\Rightarrow \mathcal{G}(t_0) \triangleright G^\omega && \text{Lem. 6} \\ &\Rightarrow \mathcal{G}(t_0) \triangleright G^i \text{ for some } i \in \omega && \text{Prop. 19 (3)} \\ &\Rightarrow \mathcal{G}(t_0) \triangleright_{\mathcal{H}} G && \text{Prop. 22 (3)} \end{aligned}$$

This concludes the proof of this direction.

(\Leftarrow) Suppose that $G(t_0) \triangleright_{\mathcal{H}} \mathcal{G}(u_0) \oplus G(u_1)$ and let us show that:

$$\mathcal{R}el \models \mathcal{H} \Rightarrow (t_0 \geq u_0 \vee t_1 \geq u_1)$$

Note that the other case is symmetric. Let $\sigma : A \rightarrow \mathcal{P}(B \times B)$ be an interpretation under which \mathcal{H} is true, and let G be its graph. We distinguish two cases. We have either:

$$\forall x, y \in B, \quad \mathcal{G}(u_1) \not\triangleright (x, G, y)$$

In this case, by Lemma 6, there is no pair (x, y) such that $(x, y) \in \sigma(u_1)$, hence $\sigma(t_1) \supseteq \sigma(u_1)$ is vacuously true.

Suppose now that there is x_1 and y_1 in B such that $\mathcal{G}(u_1) \triangleright (x_1, G, y_1)$. Notice the following:

$$\forall x, y \in B, \quad \mathcal{G}(u_0) \triangleright (x, G, y) \Rightarrow \mathcal{G}(u_0) \oplus \mathcal{G}(u_1) \triangleright (x, G, y)$$

By using Lemma 6 and this remark, we get that if $\sigma(u_0) \ni (x, y)$ then $\mathcal{G}(t_0) \triangleright_{\mathcal{H}} (x, G, y)$. By Lemma 15, and since the inequations \mathcal{H} are true under σ , we have that $\mathcal{G}(t_0) \triangleright (x, G, y)$. Hence, by Lemma 6, we get $\sigma(t_0) \ni (x, y)$ which concludes the proof. ◀

As every universal sentence can be written as the conjunction of some generalized Horn sentences, Theorem 25 gives us a characterization of the validity of all universal sentences.

5 Conclusion

We end this paper by some concluding remarks and open problems.

By characterizing the universal theory of relations, we characterized also their existential theory. Now, can we characterize the full first-order theory of relations using graphs and homomorphisms?

Another direction of work is to extend the syntax of terms. For instance, we could add the operations of union and Kleene star. In this case, terms are interpreted, not by a single graph as we did here, but by a set of graphs as in [4, Def. 4]. Graph homomorphism is generalized to the relation \blacktriangleright between sets of graphs as follows:

$$C \blacktriangleright D \quad \Leftrightarrow \quad \forall H \in D, \exists G \in C, G \triangleright H$$

With these interpretations, Theorem 7 can be easily adapted when union is added to the syntax. However, it is not clear how to adapt it in the presence of the Kleene star. Theorem 12 seems hard to adapt both for the union and the Kleene star extensions.

Even if Theorems 12 and 25 do not give decidability for the corresponding theories, we can wonder whether it can be obtained under some restrictions on the hypothesis \mathcal{H} . For instance, is it the case when the hypothesis \mathcal{H} form a Noetherian rewriting system?

We can easily adapt this work to the realm of conjunctive queries. Indeed, terms can be replaced by conjunctive queries and inequations between terms by equivalence between conjunctive queries $Q_1 \equiv Q_2$. For example, by adapting Theorem 7 we get the decidability of the following problem:

Input: Conjunctive queries Q_1, Q_2, Q_3 and Q_4 .
Output: Do we have $(Q_1 \equiv Q_2) \vee (Q_3 \equiv Q_4)$?

which generalizes the result of Chandra and Merlin [5].

References

- 1 H. Andr eka and D.A. Bredikhin. The equational theory of union-free algebras of relations. *au*, 33(4):516–532, 1995. doi:10.1007/BF01225472.
- 2 Hajnal Andr eka and Szabolcs Mikul as. Axiomatizability of positive algebras of binary relations. *Algebra universalis*, 66(7), 2011. An erratum appears at <http://www.dcs.bbk.ac.uk/~szabolcs/AM-AU-err6.pdf>. doi:10.1007/s00012-011-0142-3.
- 3 D. A. Bredikhin. The equational theory of relation algebras with positive operations. *Izv. Vyssh. Uchebn. Zaved. Mat.*, 37(3):23–30, 1993. In Russian. URL: <http://mi.mathnet.ru/ivm4374>.
- 4 Paul Brunet and Damien Pous. Petri automata for kleene allegories. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 68–79. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.17.
- 5 Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90. ACM, 1977. doi:10.1145/800105.803397.
- 6 Martin D. Davis. *Computability and Unsolvability*. McGraw-Hill Series in Information Processing and Computers. McGraw-Hill, 1958.
- 7 P. Freyd and A. Scedrov. *Categories, Allegories*. nh, 1990.
- 8 Wilfrid Hodges. *A Shorter Model Theory*. Cambridge University Press, USA, 1997.
- 9 Ian Hodkinson and Szabolcs Mikul as. Axiomatizability of reducts of algebras of relations. *Algebra Universalis*, 43:127–156, 2000. doi:10.1007/s000120050150.

- 10 Roger Maddux. *Relation Algebras*. Elsevier, 2006.
- 11 Szabolcs Mikulás. Axiomatizability of algebras of binary relations. In *Classical and New Paradigms of Computation and their Complexity Hierarchies*, pages 187–205. Springer Netherlands, 2004. doi:10.1007/978-1-4020-2776-5_11.
- 12 Donald Monk. On representable relation algebras. *The Michigan mathematical journal*, 31(3):207–210, 1964. doi:10.1307/mmj/1028999131.
- 13 Yoshiki Nakamura. Partial derivatives on graphs for kleene allegories. In *Lics*, pages 1–12. iee, 2017. doi:10.1109/LICS.2017.8005132.
- 14 A. Tarski. On the calculus of relations. *J. of Symbolic Logic*, 6(3):73–89, 1941.
- 15 A. Tarski and S. Givant. *A Formalization of Set Theory without Variables*, volume 41 of *Colloquium Publications*. ams, Providence, Rhode Island, 1987.


Co-Degeneracy and Co-Treewidth: Using the Complement to Solve Dense Instances

Gabriel L. Duarte ✉

Institute of Computing, Fluminense Federal University, Niterói, Brazil

Mateus de Oliveira Oliveira ✉ 

Department of Informatics, University of Bergen, Norway

Uéverton S. Souza¹ ✉ 

Institute of Computing, Fluminense Federal University,, Niterói, Brazil

Abstract

Clique-width and treewidth are two of the most important and useful graph parameters, and several problems can be solved efficiently when restricted to graphs of bounded clique-width or treewidth. Bounded treewidth implies bounded clique-width, but not vice versa. Problems like LONGEST CYCLE, LONGEST PATH, MAXCUT, EDGE DOMINATING SET, and GRAPH COLORING are fixed-parameter tractable when parameterized by the treewidth, but they cannot be solved in FPT time when parameterized by the clique-width unless $FPT = W[1]$, as shown by Fomin, Golovach, Lokshtanov, and Saurabh [SIAM J. Comput. 2010, SIAM J. Comput. 2014]. For a given problem that is fixed-parameter tractable when parameterized by treewidth, but intractable when parameterized by clique-width, there may exist infinite families of instances of bounded clique-width and unbounded treewidth where the problem can be solved efficiently. In this work, we initiate a systematic study of the parameters co-treewidth (the treewidth of the complement of the input graph) and co-degeneracy (the degeneracy of the complement of the input graph). We show that LONGEST CYCLE, LONGEST PATH, and EDGE DOMINATING SET are FPT when parameterized by co-degeneracy. On the other hand, GRAPH COLORING is para-NP-complete when parameterized by co-degeneracy but FPT when parameterized by the co-treewidth. Concerning MAXCUT, we give an FPT algorithm parameterized by co-treewidth, while we leave open the complexity of the problem parameterized by co-degeneracy. Additionally, we show that PRECOLORING EXTENSION is fixed-parameter tractable when parameterized by co-treewidth, while this problem is known to be $W[1]$ -hard when parameterized by treewidth. These results give evidence that co-treewidth is a useful width parameter for handling dense instances of problems for which an FPT algorithm for clique-width is unlikely to exist. Finally, we develop an algorithmic framework for co-degeneracy based on the notion of Bondy-Chvátal closure.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases FPT, treewidth, degeneracy, complement graph, Bondy-Chvátal closure

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.42

Funding *Mateus de Oliveira Oliveira*: Bergen Research Foundation, and by the Research Council of Norway (Grant Number: 288761).

Uéverton S. Souza: Grant E-26/203.272/2017 - Carlos Chagas Filho Research Support Foundation (FAPERJ), and Grant 309832/2020-9 - National Council for Scientific and Technological Development (CNPq).

¹ Corresponding author



1 Introduction

Treewidth and *clique-width* are two of the most important and useful graph parameters. Families of graphs of bounded treewidth include cactus graphs, outerplanar graphs, series-parallel graphs, Halin graphs, Apollonian networks [3], and graphs of bounded branch-width [45]. Graph classes with bounded clique-width include cographs [10], distance-hereditary graphs [35], and graphs of bounded treewidth [20]. Additionally, the clique-width of a graph is asymptotically equivalent to its rank-width [43].

An algorithmic meta-theorem due to Courcelle, Makowsky and Rotics [15] states that any problem expressible in the monadic second-order logic of graphs (MSO_1) can be solved in $f(cw) \cdot n$ time, i.e., it is fixed-parameter tractable when parameterized by the clique-width, cw , of the input graph. Originally this required a clique-width expression as part of the input. This restriction was removed when Oum and Seymour [43] gave an FPT algorithm, parameterized by the clique-width of the input graph, that finds a $2^{\mathcal{O}(cw)}$ -approximation of an optimal clique-width expression. In addition, Courcelle [16] states that any problem expressible in the monadic second-order logic of graphs with edge set quantifications (MSO_2) can be solved in time $f(tw) \cdot n$, where tw is the treewidth of the input graph. Clearly, every MSO_1 property is also a MSO_2 property. However, there are MSO_2 properties like “ G has an Hamiltonian cycle” that are not MSO_1 expressible [18]. These results have been extended a number of times [1, 9, 15, 37]. In particular, the MSO meta-theorems mentioned above were extended to LinEMSO by allowing the expressibility of optimization problems concerning maximum or minimum sets (LinEMSO properties are equivalent to MSO properties for optimization problems which can be expressed as searching for sets of vertices/edges that are optimal concerning some linear evaluation functions) [1, 17, 15]. From these meta-theorems, it follows that several problems can be efficiently solved when restricted to graphs of bounded clique-width or treewidth. Many optimization problems are LinEMSO_2 -expressible, but GRAPH COLORING , the problem of determining the chromatic number of the input graph is not a LinEMSO problem [41]. However, on graphs G with bounded treewidth, the chromatic number of G is also bounded; therefore, in this specific case, the problem is Turing-reducible to a MSO_1 problem (k - COLORING for fixed k).

Bounded treewidth implies bounded clique-width [13] but the opposite implication is not valid, as in the case of complete graphs. On the other hand, LinEMSO_2 is more expressive than LinEMSO_1 , and there exist LinEMSO_2 -expressible problems like MAXCUT , LONGEST CYCLE , LONGEST PATH and $\text{EDGE DOMINATING SET}$ that cannot be FPT when parameterized by clique-width [28, 29, 30, 31], unless $\text{FPT} = \text{W}[1]$. Additionally, GRAPH COLORING is also an FPT problem concerning treewidth parameterization that cannot be FPT when parameterized by clique-width, unless $\text{FPT} = \text{W}[1]$, see [30].

For problems that are fixed-parameter tractable when parameterized by treewidth, but intractable when parameterized by clique-width, the identification of tractable classes of instances of bounded clique-width and unbounded treewidth becomes a fundamental quest. The goal of this work is to show that *co-treewidth*, that is to say, the treewidth of the complement of the input graph, is a parameter that fulfills this quest. More precisely, we will show that several natural problems that are unlikely to be in FPT when parameterized by clique-width can be solved in FPT time when parameterized by co-treewidth. Examples of such problems are LONGEST PATH , LONGEST CYCLE , MAXCUT , $\text{EDGE DOMINATING SET}$, and GRAPH COLORING . In addition, since bounded treewidth implies bounded degeneracy (the degeneracy of a graph is upper bounded by its treewidth), we also consider the degeneracy of the complement graph, called *co-degeneracy*, as a parameter.

Let us say that a parameter x is weaker than parameter y , and y stronger than x , if the set of graph classes where x is bounded is a subset of those where y is bounded. In 2016, Saether and Telle [46] considered a graph parameter called *sm-width* which is stronger than *treewidth* and weaker than *clique-width*. They showed that *MAXCUT*, *GRAPH COLORING*, *HAMILTONIAN CYCLE* and *EDGE DOMINATING SET* are FPT when parameterized by *sm-width*. However, *co-treewidth* and *sm-width* are incomparable since trees have bounded *sm-width* but unbounded *co-treewidth*, and the complements of paths have bounded *co-treewidth* but unbounded *sm-width*. Also, note that *neighborhood diversity* [40], *twin-cover* [33], *shrub-depth* [34], and *modular-width* [32] are all weaker than *clique-width*, but none of them are stronger than *co-treewidth*. Gajarský, Lampis, and Ordyniak [32] showed that *GRAPH COLORING* and *HAMILTONIAN PATH* are $W[1]$ -hard parameterized by *shrub-depth* but FPT parameterized by *modular-width* (which is stronger than *neighborhood diversity* and *twin-cover*), they also leave as an open problem the complexity of *MAXCUT* and *EDGE DOMINATING SET* parameterized by *modular-width*. Besides, also in the context “between *treewidth* and *clique-width*”, Eiben, Ganian, Hamm, and Kwon [26] develop hybrid parameters (\mathcal{H} -*treewidth*) combining advantages of *treewidth* and modulators, the aim of \mathcal{H} -*treewidth* is to capture the *treewidth* of a modulator to the class \mathcal{H} (see [26]).

In 2016, Dvořák, Knop, and Masařík [25] showed that *k-PATH COVER* is FPT when parameterized by the *treewidth* of the complement of the input graph (i.e., the *co-treewidth* of the input). This implies that *HAMILTONIAN PATH* is FPT when parameterized by *co-treewidth*. In 2017, Knop, Koutecký, Masařík, and Toufar [38] asked about the complexity of deciding graph problems Π on the complement of G considering a parameter p of G (i.e., with respect to $p(G)$), especially for sparse graph parameters such as *treewidth*. In this paper, by showing that *LONGEST PATH*, *LONGEST CYCLE*, *MAXCUT*, *EDGE DOMINATING SET*, and *GRAPH COLORING* are all FPT when parameterized by *co-treewidth*, we exemplify that *co-treewidth* is a useful width parameter for dealing with problems for which an FPT algorithm for *clique-width* is unlikely. Besides, to the best of our knowledge, this is the first work dealing with *co-degeneracy* parameterization.

It is also natural to consider the *clique-width* of the complement graph as parameter, however, Courcelle and Olariu [20] proved that for every graph G we have $cw(\bar{G}) \leq 2 \cdot cw(G)$. Therefore, the notion of “*co-clique-width*” is redundant from the point of view of parameterized complexity. On the other hand, the notion of *co-treewidth* seems to be interesting given that bounded *co-treewidth* implies bounded *clique-width*; and *treewidth* and *co-treewidth* are incomparable parameters. Moreover, although *co-degeneracy* is incomparable with *clique-width*, it is stronger than *co-treewidth* and a useful parameter for handling some problems on dense instances. In this paper, we show that *LONGEST PATH*, *LONGEST CYCLE*, and *EDGE DOMINATING SET* are FPT when parameterized by *co-degeneracy*, while *GRAPH COLORING* is para-NP-hard. The complexity of *MAXCUT* parameterized by *co-degeneracy* is left open.

Finally, we also remark that for some graph problems, *co-treewidth* can be a parameter more useful than *treewidth*. For instance, *EQUITABLE COLORING* and *PRECOLORING EXTENSION* are well-known $W[1]$ -hard problems concerning *treewidth*; however, we remark that both problems are fixed-parameter tractable when parameterized by *co-treewidth*.

1.1 Preliminaries

We use standard graph-theoretic and parameterized complexity notation, and we refer the reader to [21, 24] for any undefined notation.

The *degeneracy* of a graph G is the least k such that every induced subgraph of G contains a vertex with degree at most k . Equivalently, the *degeneracy* of G is the least k such that its vertices can be arranged into a sequence so that each vertex is adjacent to most

k vertices preceding it in the sequence. We denote by $\text{co-deg}(G)$ the co-degeneracy of G , i.e., the degeneracy of \overline{G} . Also, for a graph G , we denote by $\text{co-tw}(G)$ the co-treewidth of G , i.e., the treewidth of \overline{G} . For short, we use *co-deg* and *co-tw* whenever the graph G is implicit.

In general, for a tree decomposition $(T, \{X_t\}_{t \in V(T)})$ it is common to distinguish one vertex r of T which will be the root of T . This introduces natural parent-child and ancestor-descendant relations in the tree T [21]. It is useful to design dynamic programming algorithms based on tree decompositions to obtain rooted tree decompositions that satisfy some auxiliary conditions. Such decompositions are so-called *nice tree decompositions*.

► **Definition 1.** A tree decomposition $(T, \{X_t\}_{t \in V(T)})$ with root node X_r is nice, if the following conditions are satisfied:

- $X_r = \emptyset$; (the root is an empty bag of T)
- If X_ℓ is a leaf node of T , then $X_\ell = \emptyset$; (each leaf X_ℓ is an empty bag of T)
- Every non-leaf node of T is of one of the following three types:
 1. Introduce node: a node t with exactly one child t' such that $X_t = X_{t'} \cup \{v\}$ for some vertex $v \notin X_{t'}$; we say that v is introduced at t .
 2. Forget node: a node t with exactly one child t' such that $X_t = X_{t'} \setminus \{w\}$ for some vertex $w \in X_{t'}$; we say that w is forgotten at t .
 3. Join node: a node t with two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.

Let G be a graph and let $(T, \{X_t\}_{t \in V(T)})$ be a nice tree-decomposition of the graph G . For each node t of T , we denote by T_t the subtree of T rooted by t . Also, we denote by G_t the subgraph of G induced by the set of vertices contained in some bag of T_t .

Based on the following results, we can assume that we are given a nice tree decomposition of G without loss of generality.

► **Theorem 2** ([5]). There exists an algorithm that, given an n -vertex graph G and an integer k , runs in time $2^{O(k)} \cdot n$ and either outputs that the treewidth of G is larger than k , or constructs a tree decomposition of G of width at most $5k + 4$.

► **Lemma 3** ([21]). Given a tree decomposition $(T, \{X_t\}_{t \in V(T)})$ of G of width at most k , one can in time $O(k^2 \cdot \max(|V(T)|, |V(G)|))$ compute a nice tree decomposition of G of width at most k that has at most $O(k \cdot |V(G)|)$ nodes.

The *clique-width* of a graph is another parameter that we will mention very often, and therefore, we briefly define this parameter for completeness. Given a graph G , the clique-width of G , denoted by $\text{cw}(G)$, is defined as the minimum number of labels needed to construct G , using the following four operations: create a single vertex v with an integer label ℓ (denoted by $\ell(v)$); take the disjoint union (i.e., co-join) of two graphs (denoted by \oplus); join by an (arc) edge every vertex labeled i to every vertex labeled j for $i \neq j$ (denoted by $\eta(i, j)$); relabel all vertices with label i by label j (denoted by $\rho(i, j)$). An algebraic term representing such a construction of G and using at most k labels is a *k-expression* of G . The clique-width of G is the minimum k for which G has a k -expression.

Given a graph G and a vertex $v \in V(G)$, $N(v)$ denotes the (open) neighborhood of v , $N[v]$ denotes the closed neighborhood of v ($N[v] = N(v) \cup \{v\}$), and $d(v)$ denotes the degree of v ($d(v) = |N(v)|$).

We say that two vertices v, w of G have the same type if $N(v) \setminus \{w\} = N(w) \setminus \{v\}$.

A graph G has neighborhood diversity at most k , if there exists a partition of $V(G)$ into at most k sets, such that all the vertices in each set have the same type. We denote by $\text{nd}(G)$ (or just nd when the graph G is implicit) the least k such that G has neighborhood diversity at most k .

A set $S \subseteq V(G)$ is a *vertex cover* of a graph G if for each edge of G at least one of its endpoints is in S . The *vertex cover number* of G , denoted by $vc(G)$, is the least k such that G has a vertex cover of size k . It is well known that if a graph G has vertex cover at most k , then $nd(G) \leq 2^k + k$ (c.f. [40]).

A *path cover* \mathcal{P} of a graph G is a set of vertex-disjoint paths of G such that each vertex in $V(G)$ belongs to a path in \mathcal{P} .

A graph G is *Hamiltonian* if there is a cycle which includes every vertex of G (such a cycle is called a *Hamiltonian cycle*). A graph G is said *k-Hamiltonian* if the deletion of at most k vertices from G results in a Hamiltonian graph.

Finally, we denote by n the number of vertices of the graph under consideration.

2 Monadic second-order logic for graphs with bounded co-treewidth

Dvořák, Knop, and Masařík [25] asked whether it is possible to extend the meta-theorem for MSO_2 for the complementary setting – i.e. to allow quantification over sets of non-edges. As shown by Courcelle, Makowsky and Rotics [15] (assuming $P \neq NP$ on unary languages), it is not possible to allow quantification over sets of edges as well as quantification over sets of non-edges, under the requirement that for the target parameter the complete graphs should have bounded width. However, as observed by Knop, Kouřtecký, Masařík, and Toufar [38], the result that k -PATH COVER² is FPT when parameterized by co-treewidth suggests that at least sometimes some extension of MSO theorem can be useful to decide properties of the complement graph. Next, by way of illustration, we show that this is precisely the case of BALANCED CO-BICLIQUE, the problem of determining the maximum integer k for which the input graph G has a pair of cliques K_1, K_2 such that $|K_1| = |K_2| = k$ and there is no edge from K_1 to K_2 . Such a pair of cliques is the complement of a balanced complete bipartite graph (balanced biclique).

Let LinEMSO_2 be the extension of LinEMSO_1 , where quantification is allowed over sets of non-edges, but quantification over sets of edges is not allowed. It is easy to see that LinEMSO_2 -expressible problems are FPT concerning co-treewidth since LinEMSO_2 -expressible problems are FPT concerning treewidth. Note that expressing a property in LinEMSO_2 is equivalent to expressing the complementary property in LinEMSO_2 .

► **Lemma 4.** *Finding the maximum balanced biclique of a graph is LinEMSO_2 -expressible.*

Proof. Given a graph G and set of vertices S_1 , it is easy to express in MSO_1 the existence of a disjoint set S_2 such that S_1 and S_2 form an induced complete bipartite subgraph: S_1 and S_2 must be independent sets, and G must contain all possible edges from S_1 to S_2 . Although comparing the cardinality of sets is not allowed in LinEMSO_2 , in this particular case, it would still be possible by verifying the existence of a perfect matching in the subgraph induced by $S_1 \cup S_2$. Thus, the problem of finding the largest S_1 meeting these conditions is LinEMSO_2 -expressible. ◀

From Lemma 4 and the LinEMSO_2 meta-theorem for optimization problems parameterized by treewidth [19], it follows that Corollary 5 holds.

► **Corollary 5.** *BALANCED CO-BICLIQUE is fixed-parameter tractable when parameterized by the co-treewidth of G .*

² k -PATH COVER is the problem of finding a path cover of size k , where k is fixed.

3 Bondy-Chvátal closure, stability and co-degeneracy

Let G be a graph with n vertices and let u and v be distinct nonadjacent vertices of G such that $d(u) + d(v) \geq n$. Ore's theorem states that G is hamiltonian if and only if $G + uv$ is hamiltonian. In 1976, Bondy and Chvátal [8] generalized Ore's theorem and defined a helpful tool: the *closure* of a graph.

Let ℓ be an integer. The $(n + \ell)$ -closure $cl_{n+\ell}(G)$ of a graph G is obtained from G by recursively adding an edge between pairs of nonadjacent vertices whose degree sum is at least $n + \ell$ until no such pair remains. Bondy and Chvátal showed that $cl_{n+\ell}(G)$ is uniquely determined by G and that G is hamiltonian if and only if $cl_n(G)$ is hamiltonian.

First, observe that the classes of graphs with bounded co-degeneracy and bounded co-treewidth are both closed under completion (edge addition), just as bounded degeneracy and treewidth are closed under edge removals. Therefore, regarding HAMILTONIAN CYCLE on graphs with co-degeneracy or co-treewidth k , without loss of generality, we can assume that $G = cl_n(G)$.

Dvořák, Knop and Masařík [25] showed that if a graph G has co-treewidth k and $G = cl_n(G)$ (that is, closed under Bondy-Chvátal closure) then G has neighborhood diversity bounded by $2^{2(k^2+k)} + 2(k^2 + k)$. Below we present some stronger results.

We call by *co-vertex cover* any set of vertices whose removal makes the resulting graph complete, i.e., a vertex cover in the complement. The *co-vertex cover number*, $co-vc(G)$, of a graph G is the minimum cardinality of a co-vertex cover in G . Recall that $co-vc(G)$ is also called *distance to clique*, and a co-vertex cover set is also called a *clique modulator*.

► **Theorem 6.** *Let $\ell \geq 0$ be an integer. If a graph G has co-degeneracy k and $G = cl_{n+\ell}(G)$ then G has co-vertex cover number bounded by $2k + \ell + 1$. In addition, a co-vertex cover of G with size at most $2k + \ell + 1$ can be found in polynomial time.*

Proof. Let G be a graph such that $G = cl_{n+\ell}(G)$ and $co-deg(G)=k$.

We may assume that G has at least $2k + \ell + 2$ vertices.

Let $v_1, v_2, v_3, \dots, v_n$ be an ordering of the vertices of \overline{G} obtained by repeatedly removing the minimum-degree vertex of \overline{G} . For each $t \in \{1, 2, \dots, n\}$ we denote by G^t the subgraph of G induced by $\{v_i : 1 \leq i \leq t\}$.

Note that G^1 is a complete graph (i.e., a K_1). Therefore, $t = 1$ is the base case.

Now, let t be an integer such that $2 \leq t \leq n$ and $|V(G) \setminus V(G^t)| \geq 2k + \ell + 1$.

Suppose by hypothesis that G^{t-1} is a complete graph. At this point, it remains to prove that G^t is also a complete graph.

Since $V(G^{t-1}) = \{v_1, \dots, v_{t-1}\}$ is a clique of G and $co-deg(G)=k$, it holds that each vertex of $V(G^{t-1})$ has degree at least $n - k - 1$ in G . Also, the vertex v_t has at least $k + \ell + 1$ neighbors in $V(G) \setminus V(G^t)$, because $|V(G) \setminus V(G^t)| \geq 2k + \ell + 1$, and by the co-degeneracy the vertex v_t has at most k non-neighbors in $V(G) \setminus V(G^t)$. Finally, as $G = cl_{n+\ell}(G)$ it holds that v_t is adjacent to all vertices of G^{t-1} in the graph G , which implies that $V(G^t)$ is also a clique of G .

Thus, in a left-right manner according the ordering, $v_1, v_2, v_3, \dots, v_n$, we can observe that each $V(G^t)$ induces a clique until meeting the first vertex v_j such that $|V(G) \setminus V(G^j)| < 2k + \ell + 1$. This implies that $\{v_j, v_{j+1}, \dots, v_n\}$ is a co-vertex cover of G with size at most $2k + \ell + 1$, and it can be computed in polynomial time. ◀

As a corollary we improve the Dvořák-Knop-Masařík bound with respect to neighborhood diversity for $\ell = 0$.

► **Corollary 7.** *Let $\ell \geq 0$ be an integer. If a graph G has co-degeneracy k and $G = cl_{n+\ell}(G)$ then G has neighborhood diversity bounded by $2^{2k+\ell+1} + 2k + \ell + 1$.*

In [25], from the fact that if $G = cl_n(G)$ then the neighborhood diversity is bounded by a function of the co-treewidth, it is claimed that HAMILTONIAN PATH is in FPT concerning co-treewidth. For an FPT algorithm for HAMILTONIAN PATH parameterized by the neighborhood diversity (nd), they point to the seminal paper of Lampis [40], which uses an algorithm of Cosmadakis and Papadimitriou [14] resulting in solving the problem in time $O^*(2^{nd \cdot \log nd})$. Recall that this strategy implies a solution for HAMILTONIAN PATH in double exponential time with respect to the co-treewidth. Below we present a much more efficient algorithm. First, we consider co-vertex cover number parameterization.

► **Lemma 8.** *LONGEST PATH and LONGEST CYCLE can be solved in time $2^{O(k \cdot \log k)} \cdot n^{O(1)}$ where k is the co-vertex cover number of the input graph.*

Proof. Let S, K be a partition of the vertices of a graph G into a co-vertex cover S and a clique K , where $|S| = k$. We assume that $|K| > 2|S|$; otherwise we can “guess”, in single-exponential time, the vertices in the longest path/cycle, so one can solve both problems using single-exponential exact algorithms for HAMILTONIAN CYCLE (or TSP), such as the Bellman–Held-Karp algorithm [2, 36].

Since $|K| > 2|S|$, there is a longest cycle and a longest path containing all vertices of K , otherwise any longest cycle/path either has no edge between two vertices of K (therefore, it has size at most $2|S|$), or it has an edge uv where $u, v \in K$, implying that there is a larger cycle/path obtained by replacing uv by a uv -path containing as internal vertices the vertices of K that were not in the cycle/path, both cases contradict the fact that the cycle/path is the longest. Also, note that LONGEST PATH can be reduced to LONGEST CYCLE by adding one universal vertex. Therefore, we focus on LONGEST CYCLE.

Now, in time $2^{O(|S|)}$ one can branch by guessing the set S_x of vertices of S that are not in the longest cycle, and then removing S_x . After that, we may assume that we are dealing with an instance G' of the HAMILTONIAN CYCLE problem, where $V' = V(G) \setminus S_x$, $G' = G[V']$, $K' = K$, and $S' = S \setminus S_x$ is a co-vertex cover of G' .

Let k' be the cardinality of S' . Recall that $k' \leq k$. Now, we branch by guessing a permutation $s_1, s_2, \dots, s_{k'}$ of the vertices of S' representing a circular order of visits of the vertices of S' in the Hamiltonian cycle C (if any). Given such a permutation $s_1, s_2, \dots, s_{k'}$, we guess the edges s_i, s_{i+1} of G' that are in C . Note that these branching steps take $O(k! \cdot 2^k)$ time. Recall that $O(k! \cdot 2^k) = 2^{O(k \log k)}$. At this point, we have guessed the set of subpaths of C induced by S' . Let P_1, P_2, \dots, P_r be the circular order of visits of such paths according to the guessed permutation.

For each pair of consecutive paths P_i, P_{i+1} either their corresponding endpoints are connected by a common neighbor in C or there is a path of vertices of K between them in such a cycle. Again, we branch by guessing in time $2^{O(k)}$ the pairs connected by a common neighbor. After that, we can construct a bipartite graph B with bipartition $V(B) = (B_1, B_2)$ where: each vertex of B_1 represents either a pair of endpoints of the paths that must have a common neighbor in C , or an endpoint that has a distinct neighbor inside K along C ; B_2 is the set of vertices of the clique K , and $E(B)$ is defined according to the edges from S' to K (for vertices representing pairs of endpoints, the neighborhood is the vertices of K that are neighbors of both endpoints). Clearly, if the guessed structures are feasible for obtaining a Hamiltonian cycle C then B has a matching of size $|B_1|$. Let M be such a matching, if any.

Since $|K| > 2|S|$ we assume that for at least one pair P_i, P_{i+1} its corresponding endpoints do not have a common neighbor in C . Hence, having the matching M and such a path cover P_1, \dots, P_r of $G'[S']$, a Hamiltonian cycle can be easily obtained for G' . Therefore, LONGEST CYCLE and LONGEST PATH can be solved in time $2^{O(k \log k)} \cdot n^{O(1)}$, where $k = co-deg(G)$. ◀

Since HAMILTONIAN PATH can be reduced to HAMILTONIAN CYCLE by adding a universal vertex, from Bondy-Chvátal Theorem, Theorem 6 and Lemma 8, the following holds.

► **Corollary 9.** HAMILTONIAN CYCLE and HAMILTONIAN PATH can be solved in time $2^{O(k \log k)} \cdot n^{O(1)}$, where k is the co-degeneracy of the input graph.

In order to extend the previous result to PATH COVER, LONGEST CYCLE, and LONGEST PATH as well as considering the same strategy for other properties, it becomes necessary to introduce the notion of stability.

3.1 The stability of a property

A property P defined on all graphs of order n is said to be $(n + \ell)$ -stable if for any graph G of order n that does not satisfy P , the fact that uv is not an edge of G and that $G + uv$ satisfies P implies $d(u) + d(v) < n + \ell$. In other words, if $uv \notin E(G)$, $d(u) + d(v) \geq n + \ell$ and $G + uv$ has property P , then G itself has property P (c.f. [12]). We denote by $s(P)$ the smallest integer $n + \ell$ such that P is $(n + \ell)$ -stable, and call it the *stability* of P .

Note that if a graph property P is $(n + \ell)$ -stable then edges between pair of vertices u, v such that $d(u) + d(v) \geq n + \ell$ can be added without destroying such a property P .

Our co-degeneracy+closure+co-vertex cover framework is based on the following facts:

1. If a property P is $(n + \ell)$ -stable and $cl_{n+\ell}(G)$ satisfies P , then G itself satisfies P .
2. If a property P is $(n + \ell)$ -stable, regarding the problem of recognizing property P parameterized by co-degeneracy we can assume without loss of generality that $G = cl_{n+\ell}(G)$.
3. If $G = cl_{n+\ell}(G)$ and G has co-degeneracy k then G has a co-vertex cover of size at most $2k + \ell + 1$. In particular, we are interested in cases where ℓ is bounded by a function of k .
4. Many problems are easily solved in FPT-time concerning co-vertex cover parameterization.

Next, we list the stability of some graph properties P (see [8, 11, 12]):

Longest Cycle: “ G has circumference k ” satisfies $s(P) = n$. (Thm. 4 in [11])

Longest Path: “ G contains a P_k ” satisfies $s(P) = n - 1$ for $4 \leq k \leq n$. (Thm. 2.40 in [12])

Path Cover: “ G has a path cover of size at most k ” satisfies $s(P) = n - k$. (Thm. 9.13 in [8])

k -Hamiltonian: “ G is k -Hamiltonian” satisfies $s(P) = n + k$. (Thm. 2.25 in [12])

At this point, it is easy to see that Corollary 10 holds.

► **Corollary 10.** LONGEST CYCLE/PATH can be solved in time $2^{O(\text{co-deg} \log \text{co-deg})} \cdot n^{O(1)}$, and k -Hamiltonian graphs can be recognized in time $2^{O((\text{co-deg}+k) \log(\text{co-deg}+k))} \cdot n^{O(1)}$.

Proof. By the stability of the properties regarding the computation of the longest cycle, longest path, and recognition of k -Hamiltonian graphs, it holds that one can assume that G is closed under Bondy-Chvátal closure for an appropriated integer ℓ ($G = cl_{n+\ell}(G)$). By Theorem 6, it holds that G has a co-vertex cover S of size $O(\text{co-deg})$, or $O(\text{co-deg} + k)$ in the k -Hamiltonian case, and such a co-vertex cover can be obtained in polynomial time.

Given a co-vertex cover S of G , in time $2^{O(|S|)}$ one can “guess” the vertices of S that must be removed or are not in the longest cycle/path. After that, the proof follows as in Lemma 8. ◀

Next, we deal with the PATH COVER problem.

► **Corollary 11.** PATH COVER can be solved in time $2^{O(\text{co-deg} \log \text{co-deg})} \cdot n^{O(1)}$.

Proof. By the stability of the property P of having a path cover of size at most k (see [8]), without loss of generality, we can assume $G = cl_{n-k}(G)$. Note that $E(G) \subseteq E(cl_n(G)) \subseteq E(cl_{n-k}(G))$. Thus, from Theorem 6 it holds that G has co-vertex cover number at most $2k+1$, where $k = co-deg(G)$. In addition, such a co-vertex cover S can be found in polynomial time. Thus, G has a path cover of size at most $2k+2$, because one can use a trivial path for each vertex of the co-vertex cover, and a single path for the remaining vertices (they induce a clique). So, the path cover number of a graph G is bounded by $2k+2$. At this point, it is enough to determine the least $r \in [1, 2k+2]$ for which G has an r -path cover (path cover of size r). Since one can reduce the problem of finding an r -path cover to HAMILTONIAN PATH by adding $r-1$ universal vertices, and the addition of universal vertices preserve the co-vertex cover number of the input graph, by Corollary 9 it holds that PATH COVER can also be solved in time $2^{O(k \log k)} \cdot n^{O(1)}$, where $k = co-deg(G)$. ◀

4 Edge Dominating Set parameterized by co-degeneracy

An edge dominating set of a graph G is a set $Q \subseteq E(G)$ such that every edge of G is either in Q or incident to at least one edge of Q . The EDGE DOMINATING SET problem consists of determining the size of a minimum edge dominating set.

In [30], Fomin et al. showed that EDGE DOMINATING SET parameterized by clique-width is W[1]-hard. In [29, 31], they showed that EDGE DOMINATING SET cannot be solved in time $f(cw) \cdot n^{o(cw)}$, unless ETH fails.

In this section, we present a single-exponential FPT algorithm for EDGE DOMINATING SET parameterized by the co-degeneracy.

▶ **Theorem 12.** EDGE DOMINATING SET can be solved in time $2^{O(co-deg)} \cdot n^{O(1)}$.

Proof. First, we observe the following key property.

▷ **Claim 13.** The problem of finding a minimum edge dominating set is equivalent to finding the smallest vertex cover S such that $G[S]$ contains a perfect matching.

Proof. Yannakakis and Gavril [49] showed that given a minimum edge dominating set Q of G , one could find a minimum maximal matching with $|Q|$ edges. Since every maximal matching is an edge dominating set, the size of a minimum edge dominating set equals the size of a minimum maximal matching.

Now, let Q be a minimum edge dominating set that is also a minimum maximal matching. Since Q is a maximal matching then $V(Q)$ is a vertex cover inducing a graph having perfect matching. In addition, there is no vertex cover smaller than $V(Q)$ that also induces a graph having perfect matching; otherwise, Q is not a minimum maximal matching.

Conversely, let S be the smallest vertex cover S of G such that $G[S]$ contains a perfect matching. Let Q be such a perfect matching of $G[S]$. Since S is a vertex cover of G then $V \setminus S$ is an independent set, which implies that Q is a maximal matching of G . In addition, Q must be a minimum maximal matching, otherwise using the previous argument we obtain a vertex cover smaller than S also having a perfect matching, a contradiction. ◀

Now, recall that enumerating vertex covers in G is the same as enumerating independent sets in the same graph, which is equivalent to enumerating the cliques of \overline{G} .

Since \overline{G} has degeneracy k , we can enumerate all cliques containing some vertex of degree at most k (such a vertex must exist and there are at most 2^k cliques containing it); by deleting this vertex and continuing the enumeration in the remaining graph, we can enumerate every clique of \overline{G} .

Therefore, if a graph G has co-degeneracy k , then G has at most $2^k \cdot n$ distinct vertex covers which we can enumerate in time $O(2^k \cdot n^{O(1)})$. Thus, by checking the existence of perfect matchings, one can find the minimum edge dominating set in time $2^k \cdot n^{O(1)}$. ◀

5 MaxCut parameterized by co-treewidth

A cut $[S, V \setminus S]$ of a graph $G = (V, E)$ is a partition of V into two subsets S and $V \setminus S$. The *size* of the cut $[S, V \setminus S]$ is the number of edges crossing it, i.e., the cardinality of the cut-set $\{uv \in E \mid u \in S, v \in V \setminus S\}$. In the MAXCUT problem, we are given an unweighted undirected graph $G = (V, E)$, and our goal is to find a cut of maximum size.

From the parameterized complexity point of view, to determine if G has a cut of size at least k is fixed-parameter tractable when parameterized by either k or $k - \lfloor \frac{|E|}{2} \rfloor$ (c.f. [42, 44]). In addition, in 2000, Bodlaender and Jansen [6] showed that MAXCUT can be solved in FPT time when parameterized by the treewidth, and, in 2013, Bodlaender, Bonsma, and Lokshantov presented an $O(2^{tw} \cdot n)$ time algorithm for the problem, where tw is the treewidth of the input graph. On the other hand, Fomin, Golovach, Lokshantov and Saurabh [29, 31] showed that MAXCUT cannot be solved in time $f(cw) \cdot n^{o(cw)}$, unless ETH fails, where cw is the clique-width of the input graph G .

Regarding co-degeneracy parameterization we left the complexity of MAXCUT open. To the best of our knowledge, the complexity of MAXCUT is unknown even for co-planar graphs (the class of planar graphs is a subclass of the 5-degenerate graphs).

Concerning co-treewidth, it is not clear whether MAXCUT can be expressed in LinEMSO_2 . Given a cut $[S, V \setminus S]$ of a graph G , the complement of the cut-set of $[S, V \setminus S]$ is the set of non-edges that have one endpoint in each subset of the partition. The main challenge to express MAXCUT using just quantification over sets of vertices and sets of non-edges is that the size of $[S, V \setminus S]$ is given by $(|S| \cdot |V \setminus S|) - |\{uv \notin E \mid u \in S, v \in V \setminus S\}|$, thus, the natural objective function is not linear, and it is not appropriate to express the problem in LinEMSO . However, using a nice tree decomposition of the complement graph, we can find the maximum cut of the input graph in single-exponential time concerning the co-treewidth.

Given a graph G and a nice tree-decomposition $(T, \{X_t\}_{t \in V(T)})$ of G , our goal is to use $(T, \{X_t\}_{t \in V(T)})$ in order to find a maximum cut of \overline{G} . Recall that for each node t of T , we denote by G_t the subgraph of G induced by the set of vertices contained in some bag of T_t . Also, for each X_t the set of forgotten vertices in G_t is denoted by F_t ($F_t = V(G_t) \setminus X_t$).

Given a cut $[S, V \setminus S]$, we say that S is left part of the partition defined by the cut, and $V \setminus S$ is the right part.

Let $C[t, S, \ell]$ be the size of a maximum cut of the subgraph $\overline{G_t}$, where S are the vertices of X_t to the left part of the partition defined by the cut, and $X_t \setminus S$ are the vertices of X_t on the right part. Also, ℓ represents how many forgotten vertices are in the left part of this cut.

At this point, it is sufficient to show how to compute in a bottom-up manner the entries of the matrix (which is regarding \overline{G} (not G)) according to each type of node of the nice tree decomposition of G . Note that the maximum size of a cut in \overline{G} equals $\max_{0 \leq \ell \leq n} \{C[r, \emptyset, \ell]\}$, where r is the root of the tree decomposition.

For each node t of T we denote by t' and t'' the children of t (if any), and for each bag X_t we denote by $E(X_t)$ the set of edges with both endpoints in X_t . Thus, we proceed as follows:

Leaf: Since $X_t = \emptyset$, it holds that $S = \emptyset$, $\ell = 0$, and $C[t, S, \ell] = 0$.

Introduce vertex: $X_t = X_{t'} \cup \{v\}$. Recall that we are working with the tree decomposition of G , so the size of cuts of \overline{G} is given by non-edges of G .

$$C[t, S, \ell] = \begin{cases} C[t', S \setminus \{v\}, \ell] + |X_t \setminus S| - |\{uv \in E(X_t) : u \notin S\}| + (|F_{t'}| - \ell) & \text{if } v \in S \\ C[t', S, \ell] + |S| - |\{uv \in E(X_t), u \in S\}| + \ell & \text{if } v \notin S \end{cases} \quad (1)$$

Note that $|F_{t'}| - \ell$ is the number of vertices forgotten to the right part of the cut.

Forget vertex: $X_t = X_{t'} \setminus \{v\}$. In this case, we take the best of two possibilities: v in the left or right part of the cut.

$$C[t, S, \ell] = \max\{C[t', S \cup \{v\}, \ell - 1], C[t', S, \ell]\} \quad (2)$$

Join: $X_t = X_{t'} = X_{t''}$. In this case we have to do the union of two partial solutions. Since non-edges of $G[X_t]$ are non-edges of both $G_{t'}$ and $G_{t''}$, they must not be counted twice. In addition, there are non-edges between forgotten vertices of $G_{t'}$ and $G_{t''}$ that must be counted, so:

$$C[t, S, \ell] = \max_{0 \leq i \leq \ell} \{C[t', S, i] + C[t'', S, \ell - i] - (|S| \cdot |X_t \setminus S| - |\{uv \in E(X_t) : u \in S, v \notin S\}|) + i \cdot (|F_{t''}| - (\ell - i)) + (|F_{t'}| - i) \cdot (\ell - i)\} \quad (3)$$

Since the correctness of the recurrences is straightforward, the matrix has size $2^{\mathcal{O}(\text{co-tw})} \cdot n^2$, and each entry can be computed in time $\mathcal{O}(n)$, the following theorem holds.

► **Theorem 14.** *MAXCUT can be solved in $2^{\mathcal{O}(\text{co-tw})} \cdot n^3$.*

6 Treewidth vs. co-treewidth

In the previous sections we showed that PATH COVER, LONGEST CYCLE, LONGEST PATH, MAXCUT, and EDGE DOMINATING SET are all FPT concerning co-treewidth parameterization. These results contrast with the intractability of such problems regarding clique-width parameterization. Since all of these problems are also fixed-parameter tractable when parameterized by treewidth, it becomes interesting to identify problems that are tractable for co-treewidth but intractable concerning treewidth, as well as the opposite.

The TSP problem is NP-hard on complete graphs (co-treewidth equal to zero) but fixed-parameter tractable when parameterized by treewidth [4]. On the other hand, Fellows et al. [27] showed that PRECOLORING EXTENSION and EQUITABLE COLORING are W[1]-hard when parameterized by treewidth; next, we contrast these results by remarking that both problems are fixed-parameter tractable using co-treewidth as the parameter.

6.1 Coloring and covering problems

Each color class of a proper coloring of a graph G is an independent set, i.e., each color class is a clique in the complement. So, to solve GRAPH COLORING working with the complement graph, we must solve CLIQUE COVER in \overline{G} . Therefore, GRAPH COLORING parameterized by co-degeneracy/co-treewidth is equivalent to CLIQUE COVER parameterized by degeneracy/treewidth. It is known that GRAPH COLORING and CLIQUE COVER are NP-hard on planar graphs [39, 22]. Thus, they are para-NP-hard with respect to co-degeneracy.

Regarding co-treewidth, it is not clear if CLIQUE COVER can be solved using the $MSOL_2$ framework because in contrast with GRAPH COLORING, the size of the solution is unbounded on bounded treewidth graphs. However, in [47], van Rooij, Bodlaender, van Leeuwen, Rossmanith, and Vatshelle present an FPT algorithm for γ -CLIQUE COVER parameterized by treewidth. The γ -CLIQUE COVER problem is a generalization of CLIQUE COVER where the goal is to find a minimum collection C of disjoint cliques covering $V(G)$ such that the size of every clique in C is contained in γ (a set of integers). They showed a FPT algorithm that computes the size and number of minimum γ -clique covers of G . Preliminary parts of [47] have appeared in [7, 48]. Thus, GRAPH COLORING is FPT concerning co-treewidth.

The EQUITABLE COLORING problem is a variation of GRAPH COLORING where we are asked to find the minimum integer k for which the input graph G admits a proper k -coloring such that the sizes of any two color classes differ by at most one. Again, to solve EQUITABLE COLORING one can consider the complementary problem, i.e., EQUITABLE CLIQUE COVER parameterized by treewidth. In the EQUITABLE CLIQUE COVER problem, we are asked to find the minimum integer k such that the input graph G admits a clique cover of size k such that the sizes of any two cliques of the cover differ by at most one. Thus, one can use the algorithm for γ -CLIQUE COVER to solve EQUITABLE CLIQUE COVER, considering that all the cliques must have size either ℓ or $\ell - 1$ ($\gamma = \{\ell, \ell - 1\}$). Using the folklore fact that for a graph G , every clique of G is contained in some bag of a tree decomposition of G , it follows that we only need to consider ℓ in $[2, tw(G) + 1]$. Therefore, the running time of the algorithm increases by a factor of at most $tw(G)$, and EQUITABLE COLORING can also be solved in FPT time concerning co-treewidth. Besides, Gomes, Lima and dos Santos [23], using fast subset convolution as in [47], also showed an FPT-algorithm concerning treewidth for counting clique covers of G having only cliques of size ℓ and $\ell - 1$.

6.1.1 Precoloring Extension parameterized by co-treewidth

PRECOLORING EXTENSION is a generalization of GRAPH COLORING, where we are given a graph $G = (V, E)$ with a subset $P \subseteq V$ of precolored vertices, a precoloring c_P of the vertices of P , and asked to determine the minimum integer k for which G admits a proper k -coloring c which extends c_P (that is, $c(v) = c_P(v)$ for all $v \in P$).

Again, we work with the complementary problem, which we propose to call CLIQUE COVER EXTENSION. In such a problem the input is the same as in PRECOLORING EXTENSION, and the goal is to determine the minimum size of a clique cover for which vertices with the same color are in the same clique, and no clique has a pair of precolored vertices v, u such that $c_P(v) \neq c_P(u)$.

Next, we present a standard dynamic programming based on nice tree decompositions to solve CLIQUE COVER EXTENSION parameterized by treewidth. The proposed algorithm has a single-exponential dependency on the treewidth and preserves linearity with respect to n .

First, we assume that each color class induces a clique; otherwise, there is no solution. Thus, in the forget node t of a precolored vertex v it holds that all vertices precolored with color $c_P(v)$ belong to the graph G_t .

Let $C[t, S]$ be the minimum number of cliques needed to cover the vertices of $V(G_t) \setminus S$ in G_t (the subgraph rooted by the node t) according to the constraints of CLIQUE COVER EXTENSION, where S is a subset of X_t . Since each clique is contained in some bag, we assume that each clique is formed when its last vertex is introduced. Therefore, the matrix is filled in a bottom-up manner as follows.

Leaf: $X_t = \emptyset$, thus $S = \emptyset$ and $C[t, S] = 0$.

Introduce vertex: We are introducing the vertex v and all its edges. Thus, we have two possibilities: $v \in S$ or $v \notin S$. In the second case, v forms a new clique which either has size one or is formed by v together with some non-covered neighbors in $G_{t'}$.

$$C[t, S] = \begin{cases} C[t', S \setminus \{v\}], & \text{if } v \in S \\ \min_{\substack{W \\ v \in W}} \{C[t', S \cup W]\} + 1, & \text{if } v \notin S \end{cases} \quad (4)$$

where the minimum is taken over all possible $W \subseteq N(v) \cap (X_t \setminus S)$ such that $G[W]$ is a clique (including the empty set) and:

- $W \cup \{v\}$ contains no pair of precolored vertices u, w such that $c_P(u) \neq c_P(w)$;
- if $W \cup \{v\}$ contains a precolored vertex then it contains all vertices precolored with the same color.

Forget vertex: In this node the vertex v is forgotten. Since this vertex must be covered with a clique of $G_{t'}$, we have the following

$$C[t, S] = C[t', S]$$

Join: In this case, we are joining solutions of the graphs rooted by nodes $X_{t'}$ and $X_{t''}$. Every vertex of $V(G_t) \setminus S$ should be covered by a clique in either $G_{t'}$ or $G_{t''}$. To avoid counting twice some cliques of $G[X_t]$, it is sufficient to note that if a vertex of X_t is covered in $G_{t'}$, then we can assume that it is not covered in $G_{t''}$, and vice versa. This implies that for each pair of solutions to be analyzed (one from each child), the cliques of $G[X_t]$ are considered in at most one of them.

$$C[t, S] = \min_{\substack{S', S'' \\ S = S' \cap S'', S' \cup S'' = X_t}} \{C[t', S'] + C[t'', S'']\}$$

where the minimum is taken over all possible S', S'' such that $S = S' \cap S'', S' \cup S'' = X_t$.

Since the matrix has size $\mathcal{O}(2^{tw(G)} \cdot tw(G) \cdot n)$, and each entry can be computed in $\mathcal{O}(2^{tw(G)})$, the following holds.

► **Theorem 15.** CLIQUE COVER EXTENSION can be solved in $2^{\mathcal{O}(tw(G))} \cdot n$.

► **Corollary 16.** PRECOLORING EXTENSION can be solved in $2^{\mathcal{O}(co-tw(G))} \cdot n$.

7 Concluding Remarks

LONGEST CYCLE, LONGEST PATH, PATH COVER, MAXCUT, EDGE DOMINATING SET and GRAPH COLORING are all fixed-parameter tractable when parameterized by treewidth, but they are W[1]-hard when parameterized by clique-width. To handle dense instances of problems that are hard when parameterized by clique-width, we have considered the notions of co-degeneracy and co-treewidth of a graph.

We have proposed a framework based on Bondy-Chvátal closure for working with co-degeneracy. Using this framework we showed that LONGEST CYCLE, LONGEST PATH and PATH COVER are FPT when parameterized by co-degeneracy. Additionally, using a different approach, we showed that EDGE DOMINATING SET is also FPT when parameterized by co-degeneracy. Conversely, we remark that GRAPH COLORING is para-NP-hard regarding co-degeneracy parameterization while the complexity of MAXCUT is left open. On the other hand, both GRAPH COLORING and MAXCUT are FPT when parameterized by co-treewidth.

We also have shown that PRECOLORING EXTENSION is fixed-parameter tractable taking the co-treewidth as parameter, while it is known to be W[1]-hard when parameterized by treewidth (see [27]). The same holds for EQUITABLE COLORING. In contrast, CLIQUE COVER EXTENSION and EQUITABLE CLIQUE COVER are FPT when parameterized by treewidth and W[1]-hard when parameterized by co-treewidth.

These results, which are summarized in Table 1, give evidence that co-degeneracy and co-treewidth are handy width parameters for dealing with problems for which FPT algorithms parameterized by clique-width are unlikely to exist.

■ **Table 1** Parameterized complexity concerning treewidth, co-treewidth, co-degeneracy, and clique-width of graph problems addressed in this work. Courcelle and Olariu [20] proved that for every graph G we have $cwd(\overline{G}) \leq 2 \cdot cwd(G)$, thus the W[1]-hardness of CLIQUE COVER is implied from GRAPH COLORING. Also, the indicated para-NP-hardness are inherited from GRAPH COLORING or CLIQUE COVER. The main results presented in this work are highlighted in red.

	<i>tw</i>	<i>co-tw</i>	<i>co-deg</i>	<i>cw</i>
LONGEST PATH	FPT	FPT	FPT	W[1]-h
LONGEST CYCLE	FPT	FPT	FPT	W[1]-h
EDGE DOMINATING SET	FPT	FPT	FPT	W[1]-h
MAXIMUM CUT	FPT	FPT	open	W[1]-h
GRAPH COLORING	FPT	FPT	para-NP-h	W[1]-h
CLIQUE COVER	FPT	FPT	para-NP-h	W[1]-h
PRECOLORING EXTENSION	W[1]-h	FPT	para-NP-h	W[1]-h
EQUITABLE COLORING	W[1]-h	FPT	para-NP-h	W[1]-h
CLIQUE COVER EXTENSION	FPT	W[1]-h	para-NP-h	W[1]-h
EQUITABLE CLIQUE COVER	FPT	W[1]-h	para-NP-h	W[1]-h

We remark that $\min\{\text{treewidth}, \text{co-treewidth}\}$ seems to be a nice parameter between treewidth and clique-width. Note that every problem which can be expressed in both LinEMSO_2 and LinEMSO_2 is solvable in FPT-time when parameterized by $\min\{\text{treewidth}, \text{co-treewidth}\}$. Therefore, co-treewidth is a powerful tool to manipulate dense graphs.

We left the complexity of MAXCUT parameterized by co-degeneracy as an open problem. We remark that determining the complexity of MAXCUT seems to be a challenge even for co-planar graphs. Also, investigating the applicability of co-treewidth for problems that are hard when parameterized by treewidth is an interesting research direction. In particular, the complexity of LIST COLORING parameterized by co-treewidth is another interesting question.

Finally, we note that one can also consider parameters between co-degeneracy and co-treewidth such as co-contraction degeneracy, which is defined as the maximum degeneracy of a minor of the complement of G .

References

- 1 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.
- 2 Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM (JACM)*, 9(1):61–63, 1962.
- 3 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998.
- 4 Hans L Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. In *International Colloquium on Automata, Languages, and Programming*, pages 196–207. Springer, 2013.
- 5 Hans L Bodlaender, Pål Grónås Drange, Markus S Dregi, Fedor V Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $\tilde{c}^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.
- 6 Hans L. Bodlaender and Klaus Jansen. On the complexity of the maximum cut problem. *Nordic Journal of Computing*, 7(1):14–31, 2000.
- 7 Hans L Bodlaender, Erik Jan Van Leeuwen, Johan MM Van Rooij, and Martin Vatshelle. Faster algorithms on branch and clique decompositions. In *International Symposium on Mathematical Foundations of Computer Science*, pages 174–185. Springer, 2010.
- 8 J Adrian Bondy and Vasek Chvátal. A method in graph theory. *Discrete Mathematics*, 15(2):111–135, 1976.
- 9 Richard B Borie, R Gary Parker, and Craig A Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(1-6):555–581, 1992.
- 10 Andreas Brandstädt, Feodor F Dragan, Hoàng-Oanh Le, and Raffaele Mosca. New graph classes of bounded clique-width. *Theory of Computing Systems*, 38(5):623–645, 2005.
- 11 Stephan Brandt and Henk Jan Veldman. Degree sums for edges and cycle lengths in graphs. *Journal of graph theory*, 25(4):253–256, 1997.
- 12 Hajo Broersma, Zdeněk Ryjáček, and Ingo Schiermeyer. Closure concepts: a survey. *Graphs and Combinatorics*, 16(1):17–48, 2000.
- 13 Derek G Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005.
- 14 Stavros S Cosmadakis and Christos H Papadimitriou. The traveling salesman problem with many visits to few cities. *SIAM Journal on Computing*, 13(1):99–108, 1984.
- 15 B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.
- 16 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.
- 17 Bruno Courcelle. The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues. *RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications*, 26(3):257–286, 1992.
- 18 Bruno Courcelle. The monadic second order logic of graphs VI: On several representations of graphs by relational structures. *Discrete Applied Mathematics*, 54(2-3):117–149, 1994.
- 19 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.
- 20 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.
- 21 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 22 David P Dailey. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete. *Discrete Mathematics*, 30(3):289–293, 1980.

- 23 Guilherme de C. M. Gomes, Carlos V. G. C. Lima, and Vinícius Fernandes dos Santos. Parameterized complexity of equitable coloring. *Discrete Mathematics & Theoretical Computer Science*, 21(1), 2019. URL: <http://dmtcs.episciences.org/5464>.
- 24 Rodney G. Downey and Michael R. Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.
- 25 Pavel Dvořák, Dusan Knop, and Tomáš Masarík. Anti-path cover on sparse graph classes. In Jan Bouda, Lukás Holík, Jan Kofron, Jan Strejcek, and Adam Rambousek, editors, *Proceedings 11th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2016, Telč, Czech Republic, 21st-23rd October 2016*, volume 233 of *EPTCS*, pages 82–86, 2016.
- 26 Eduard Eiben, Robert Ganian, Thekla Hamm, and O joungh Kwon. Measuring what Matters: A Hybrid Approach to Dynamic Programming with Treewidth. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.MFCS.2019.42.
- 27 Michael R. Fellows, Fedor V. Fomin, Daniel Lokshantov, Frances Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.
- 28 Fedor V Fomin, Petr A Golovach, Daniel Lokshantov, and Saket Saurabh. Clique-width: on the price of generality. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 825–834. SIAM, 2009.
- 29 Fedor V Fomin, Petr A Golovach, Daniel Lokshantov, and Saket Saurabh. Algorithmic lower bounds for problems parameterized by clique-width. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 493–502. SIAM, 2010.
- 30 Fedor V Fomin, Petr A Golovach, Daniel Lokshantov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM Journal on Computing*, 39(5):1941–1956, 2010.
- 31 Fedor V Fomin, Petr A Golovach, Daniel Lokshantov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM Journal on Computing*, 43(5):1541–1563, 2014.
- 32 Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In *International Symposium on Parameterized and Exact Computation*, pages 163–176. Springer, 2013.
- 33 Robert Ganian. Twin-cover: Beyond vertex cover in parameterized algorithmics. In *International Symposium on Parameterized and Exact Computation*, pages 259–271. Springer, 2011.
- 34 Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, Patrice Ossona de Mendez, and Reshma Ramadurai. When trees grow low: shrubs and fast MSO 1. In *International Symposium on Mathematical Foundations of Computer Science*, pages 419–430. Springer, 2012.
- 35 Martin Charles Golumbic and Udi Rotics. On the clique-width of some perfect graph classes. *International Journal of Foundations of Computer Science*, 11(03):423–443, 2000.
- 36 Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, 10(1):196–210, 1962.
- 37 Petr Hliněný, Sang-il Oum, Detlef Seese, and Georg Gottlob. Width parameters beyond tree-width and their applications. *The computer journal*, 51(3):326–362, 2008.
- 38 Dušan Knop, Martin Koutecký, Tomáš Masařík, and Tomáš Toufar. Simplified algorithmic metatheorems beyond MSO: treewidth and neighborhood diversity. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 344–357. Springer, 2017.
- 39 Daniel Král', Jan Kratochvíl, Zsolt Tuza, and Gerhard J. Woeginger. Complexity of coloring graphs without forbidden induced subgraphs. In Andreas Brandstädt and Van Bang Le, editors, *Graph-Theoretic Concepts in Computer Science*, pages 254–262, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

- 40 Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.
- 41 Clemens Lautemann. Logical definability of NP-optimisation problems with monadic auxiliary predicates. In *International Workshop on Computer Science Logic*, pages 327–339. Springer, 1992.
- 42 Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: Maxsat and maxcut. *J. Algorithms*, 31(2):335–354, 1999.
- 43 Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.
- 44 Elena Prieto. The method of extremal structure on the k -maximum cut problem. In *Proceedings of the 2005 Australasian Symposium on Theory of Computing-Volume 41*, pages 119–126. Australian Computer Society, Inc., 2005.
- 45 Neil Robertson and Paul D. Seymour. Graph minors. X. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.
- 46 Sigve Hortemo Sæther and Jan Arne Telle. Between treewidth and clique-width. *Algorithmica*, 75(1):218–253, 2016.
- 47 Johan M. M. van Rooij, Hans L. Bodlaender, Erik Jan van Leeuwen, Peter Rossmanith, and Martin Vatshelle. Fast dynamic programming on graph decompositions, 2018. [arXiv: 1806.01667](https://arxiv.org/abs/1806.01667).
- 48 Johan MM Van Rooij, Hans L Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *European Symposium on Algorithms*, pages 566–577. Springer, 2009.
- 49 Mihalis Yannakakis and Fanica Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, 1980.

Isometric Embeddings in Trees and Their Use in Distance Problems

Guillaume Ducoffe  

National Institute of Research and Development in Informatics, Bucharest, Romania
University of Bucharest, Romania

Abstract

We present powerful techniques for computing the diameter, all the eccentricities, and other related distance problems on some geometric graph classes, by exploiting their “tree-likeness” properties. We illustrate the usefulness of our approach as follows:

- We propose a subquadratic-time algorithm for computing all eccentricities on partial cubes of bounded lattice dimension and isometric dimension $\mathcal{O}(n^{0.5-\varepsilon})$. This is one of the first positive results achieved for the diameter problem on a subclass of partial cubes beyond median graphs.
- Then, we obtain almost linear-time algorithms for computing all eccentricities in some classes of face-regular plane graphs, including benzenoid systems, with applications to chemistry. Previously, only a linear-time algorithm for computing the diameter and the center was known (and an $\tilde{\mathcal{O}}(n^{5/3})$ -time¹ algorithm for computing all the eccentricities).
- We also present an almost linear-time algorithm for computing the eccentricities in a polygon graph with an additive one-sided error of at most 2.
- Finally, on any cube-free median graph, we can compute its absolute center in almost linear time. Independently from this work, Bergé and Habib have recently presented a linear-time algorithm for computing all eccentricities in this graph class (*LAGOS’21*), which also implies a linear-time algorithm for the absolute center problem.

Our strategy here consists in exploiting the existence of some embeddings of these graphs in either a system or a product of trees, or in a single tree but where each vertex of the graph is embedded in a subset of nodes. While this may look like a natural idea, the way it can be done efficiently, which is our main technical contribution in the paper, is surprisingly intricate.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Tree embeddings, Range queries, Centroid decomposition, Heavy-path decomposition, Diameter, Radius and all Eccentricities computations

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.43

Funding *Guillaume Ducoffe*: This work was supported by project PN-19-37-04-01 “New solutions for complex problems in current ICT research fields based on modelling and optimization”, funded by the Romanian Core Program of the Ministry of Research and Innovation (MCI) 2019–2022.

1 Introduction

This paper is about classic location problems on graphs and metric spaces. Although we focus on unweighted undirected graphs in what follows, it should be clear to the reader that most of our results could also be applied to discrete metric spaces. For standard graph terminology, see [12, 33]. The distance in a graph $G = (V, E)$ between two vertices $u, v \in V$ equals the minimum number of edges in a uv -path, and is denoted $d_G(u, v)$. For any vertex u in G , let $e_G(u) := \max_{v \in V} d_G(u, v)$ be its eccentricity. The diameter and the radius of G are the maximum and minimum eccentricities of a vertex. We denote the former and the latter

¹ The $\tilde{\mathcal{O}}(\cdot)$ notation suppresses poly-logarithmic factors.



© Guillaume Ducoffe;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 43; pp. 43:1–43:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

by $\text{diam}(G)$ and $\text{rad}(G)$, respectively. The center of G contains all vertices with minimum eccentricity. We study the problem of computing all eccentricities in Sec. 2. Finally, to any graph G we can associate a continuous metric space (X_G, d) : obtained from G by replacing every edge with a unit-length geodesic. Motivated by the problem of locating the “switching center” in a communication network, Hakimi’s absolute center problem consists in, being given G , computing all points of X_G that minimize their largest distance to a vertex of G [59]. We investigate this problem in Sec. 5.

Related work. There is a naive algorithm for computing all the eccentricities, resp. the absolute center, in an n -vertex m -edge graph in total $\mathcal{O}(nm)$ time. Conversely, assuming the so-called Strong-Exponential-Time Hypothesis (SETH), if an algorithm computes all the eccentricities in a graph, or even just the diameter, in $\mathcal{O}(n^a m^b)$ time, then we must have $a + b \geq 2$ [70]. In what follows, by *truly subquadratic* we mean $\mathcal{O}(n^a m^b)$ time where $a + b < 2$. The problem of finding truly subquadratic algorithms for the diameter problem on some special graph classes has been addressed in many papers [1, 14, 15, 17, 18, 23, 30, 31, 42, 46, 47, 45, 51, 55, 69]. There are comparatively much fewer results for the absolute center problem, *e.g.*, see [60, 63, 65] for previous results on trees and cacti.

As it is often the case, such problems become much easier for the graphs with a suitable “tree-like” representation such as: bounded tree-width graphs [17] and bounded clique-width graphs [44]. In the context of Metric Graph Theory (hereafter called MGT), one natural way to define tree-likeness of a graph is by using embeddings in trees. Recall that, if (X, d_X) and (Y, d_Y) are metric spaces, then an embedding is simply an injective function $\varphi : X \rightarrow Y$. Its distortion is the least $\alpha \geq 1$ s.t., for all $x, x' \in X$, we have $\alpha^{-1}d_X(x, x') \leq d_Y(\varphi(x), \varphi(x')) \leq \alpha d_X(x, x')$. The stretch is the least $\beta \geq 0$ s.t., for all $x, x' \in X$, we have $|d_X(x, x') - d_Y(\varphi(x), \varphi(x'))| \leq \beta$. An isometric embedding is one s.t. $\alpha = 1$, or equivalently $\beta = 0$. A quasi isometric embedding is one such that $\alpha = \mathcal{O}(1)$, or even better $\beta = \mathcal{O}(1)$. If we are given an isometric embedding of a graph in a tree (resp., a quasi isometric embedding), then we can solve exactly (resp., approximately) the diameter problem in linear time. Unfortunately, the graph classes to which this nice result can be applied are rather restricted. *E.g.*, the graphs that can be *isometrically* embedded in a *weighted* tree are exactly the block graphs [4, 61]. See also [5, 9] for an efficient recognition of tree metrics. More generally, all the metric spaces that embed in a tree with constant distortion have a bounded Gromov hyperbolicity: a polynomial-time computable parameter from geometric group theory that is inspired by the four-point characterization of tree metrics [58]. While many real-life networks are known to have bounded hyperbolicity [2], this is *not* the case for important classes in MGT such as: *median graphs* (even of dimension at most two), *Helly graphs* (even of strong isometric dimension at most two) and ℓ_1 -*graphs*. We refer to [3] for their respective definitions (median graphs are defined in Sec. 5).

Thus, we need to consider stronger notions of tree embeddings, or embeddings in more complicated “tree-like” spaces. Recall that the *Cartesian product* of graphs G_1, G_2 , denoted by $G_1 \square G_2$, has vertex-set $V(G_1) \times V(G_2)$ and edge-set $\{(u_1, u_2)(u_1, v_2) \mid u_2 v_2 \in E(G_2)\} \cup \{(u_1, u_2)(v_1, u_2) \mid u_1 v_1 \in E(G_1)\}$. In [21], Chepoi studied the benzenoid systems: a subclass of planar graphs with applications to chemistry, and he proved that they can be isometrically embedded in linear time in the Cartesian product of three trees. In a subsequent work [23], Chepoi et al. used this nice property in order to compute the diameter, and even the center, of these graphs in linear time. However, they also needed certain “total monotonic” property to hold for the distance-matrix of the graphs considered. To our best knowledge, the problem of computing all the eccentricities within benzenoid systems in almost linear time has been open until our work. Furthermore, we point out that the existence of similar embeddings as the one in [23], in a product or a system of constantly many trees, has been thoroughly

investigated for many graph classes [8, 6, 7, 25, 20, 35, 36, 37, 38, 39, 40, 41, 48, 68]. While it is NP-hard in general to decide whether such an embedding exists [8], we note that several of these above previous works have presented almost linear-time algorithms for this problem on some special cases.

Our results. We first prove that, for any fixed k , we can compute all eccentricities in a graph in quasi linear time if it is isometrically embedded in the Cartesian product of k trees (Theorem 1). This improves on Chepoi et al. [23] since we needn't any additional assumption on the distance-matrix and we solve a more general problem than just computing the diameter or the center. We apply this very general result to the following geometric graph classes:

- the triangular systems and hexagonal systems (*a.k.a.*, benzenoids), see Sec. 2.2;
- and the graphs of bounded lattice dimension, that can be isometrically embedded in the product of constantly many paths (with an additional technical assumption needed for computing the embedding), see Sec. 2.1.

Our actual result for Theorem 1 is even more general than what we stated above, since it can also be applied to other types of tree embeddings. If the embedding is not isometric but it has $\mathcal{O}(1)$ stretch or distortion, then our above approach leads to approximation algorithms for computing all the eccentricities. In particular,

- we prove that for any fixed k , all eccentricities in a k -polygon graph can be approximated in almost linear time within an additive one-sided error of at most 2, see Sec. 2.3.

In the second part of the paper, we consider a different type of tree embedding: from a graph G to a single tree T , but where each vertex u of G is mapped to a *subset* of nodes $U \subseteq V(T)$. This is similar to the notion of clan embedding, introduced in [52]. It leads us to solve a more general eccentricity problem on trees, that may be of independent interest for k -facility location problems [64, 73]. This is our second main technical contribution in the paper. More precisely, given an n -node tree T , the eccentricity of a node-subset U is defined as $e_T(U) = \max_{v \in V(T)} \min_{u \in U} d_T(v, u)$.

- We prove that after an $\mathcal{O}(n \log n)$ -time pre-processing, the eccentricity of any node-subset U can be computed in $\mathcal{O}(|U| \log^2 n)$ query time. See Theorem 9. These running times are optimal up to polylogarithmic factors.

Let $k \in \mathbb{N}$ and $\varepsilon \in (0; 1)$ be arbitrary. We observe that, combined to Theorem 1 in [52], our result implies a data structure for computing, for any graph G and any vertex-subset U (up to pre-processing), a $16k$ -approximation of $e_G(U)$ (resp., an $\tilde{\mathcal{O}}\left(\frac{\log n}{\varepsilon}\right)$ -approximation) in expected $\tilde{\mathcal{O}}(|U|n^{1/k})$ time (resp., in expected $\tilde{\mathcal{O}}((1 + \varepsilon) \cdot |U|)$ time). See Sec. 5.1.

We give another application of Theorem 9, in the context of distance and routing labelling schemes. Indeed, a successful approach for computing such schemes with small label sizes on geometric graph classes is as follows. Roughly, the vertices of a graph get partitioned recursively into convex subgraphs. At each step, the vertices of such convex subgraphs are projected to their respective boundary, so as to hit all shortest-paths to vertices in nearby convex subgraphs of the decomposition. Then, by mimicking the recursive construction of the labels, if the projections to the boundaries have some nice properties then, it becomes possible to compute all eccentricities, as well as to solve other related distance problems. It turns out that such embeddings are often *tree-like*, in the sense of either Theorem 1 or Theorem 9. We illustrate this with the case of cube-free median graphs (formally defined in Sec. 5). The cube-free median graphs properly contain the partial double trees (which can be isometrically embedded in the Cartesian product of two trees), and if their maximum degree is Δ then, they can be isometrically embedded in the Cartesian product of $\Delta^{\mathcal{O}(1)}$

trees [25]. However, in general they cannot be embedded in the product of constantly many trees, and so, Theorem 1 cannot be applied. We overcome this issue by relying on a recursive partition scheme as above, where the boundaries induce *isometric trees*, see [26]. Doing so, ■ we present the first almost linear-time algorithm for computing the absolute center of cube-free median graphs, see Theorem 10. Our approach can also be used in order to compute all eccentricities in this graph class.

Recently, Bergé and Habib have presented a linear-time algorithm for computing all eccentricities within the median graphs of bounded dimension (generalizing cube-free median graphs) [11]. A reviewer observed that the absolute center of a median graph is always: either its center if it is an independent set, or the middle points of all edges with their both ends in the center. As a result, we can also compute the absolute center of cube-free median graphs in linear time. That being said, we think that our alternative algorithm, although it is slightly slower, has the potential to be generalized to other graph classes beyond the median graphs. Indeed, to prove Theorem 10, we use some deep structural properties obtained recently for cube-free median graphs [26]. We point out that very similar properties have been obtained in the past for completely unrelated graph classes, such as planar graphs of non-positive combinatorial curvature [24, 29, 28]. Thus, there is room for generalizing our approach far beyond the cube-free median graphs.

Since both Theorems 1 and 9 are very general, we expect them to find applications beyond the classes studied in the paper.

Organization of the paper. In Sec. 2 we state Theorem 1, then we summarize our results obtained for subclasses of partial cubes, planar graphs and circle graphs. We postpone the proof of Theorem 1 to Sec. 3, due to its technicality. Our algorithms in Sec. 3 have an exponential dependency on the number k of trees considered. In Sec. 4, we give simple *conditional lower bounds* showing their near optimality. Finally, in Sec. 5, we address the problem of computing the eccentricity of k -subsets of nodes in a tree, and its application to the absolute center problem for cube-free median graphs.

Due to lack of space, several proofs are omitted from the following technical sections.

2 Eccentricity computation in some geometric graph classes

This section is devoted to the applications of our Theorem 1, which we formally state in what follows. We defined the Cartesian product of two graphs in Sec. 1. The *strong product* of graphs G_1, G_2 , denoted by $G_1 \boxtimes G_2$, is a supergraph of the Cartesian product with additional edge-set $\{(u_1, u_2)(v_1, v_2) \mid u_1v_1 \in E(G_1), u_2v_2 \in E(G_2)\}$. Finally, an embedding of a graph $G = (V, E)$ in a *system* of trees T_1, T_2, \dots, T_k is defined as k projections $\varphi_i : V \rightarrow V(T_i)$, $1 \leq i \leq k$. Then, the distortion of this embedding (resp. its stretch) is defined as the least α s.t., $\forall x, y \in X$, $\alpha^{-1}d(x, y) \leq \min_i d_{T_i}(\varphi_i(x), \varphi_i(y)) \leq \alpha d(x, y)$ (resp., as the least β s.t. $\forall x, y \in X$, $|d(x, y) - \min_i d_{T_i}(\varphi_i(x), \varphi_i(y))| \leq \beta$). We consider quasi-isometric embeddings of graphs and metric spaces in: Cartesian products of trees, strong products of trees, and systems of trees.

► **Theorem 1.** *Let $G = (V, E)$ be a graph and let T_1, T_2, \dots, T_k be a collection of k trees, where $N := \sum_{i=1}^k |V(T_i)|$.*

1. *If we are given an isometric embedding of G in either the system or the Cartesian product of these k trees, then we can compute all eccentricities in G in $\mathcal{O}(2^{\mathcal{O}(k \log k)}(N + n)^{1+o(1)})$ time².*

² See [17, Lemma 5] for a more careful analysis of the running time.

2. If we are given an embedding of G with distortion α (resp., with stretch β) in either the system or the Cartesian product of these k trees, then we can compute an α^2 -approximation (resp., an $+2\beta$ -approximation) of all eccentricities in G in $\mathcal{O}(2^{\mathcal{O}(k \log k)}(N+n)^{1+o(1)})$ time. It can be improved to an α -approximation (resp., a $+\beta$ -approximation) if the embedding only has one-sided distance errors.
3. All the results above also hold if we are given an embedding of G in the strong product of these k trees, with an improved runtime in $\mathcal{O}(N+kn)$.

See Sec. 3 for a sketch of the proof. We left open whether our approach could be generalized to, say, the direct product or the layered cross product [50] of constantly many trees. In what follows, we apply Theorem 1 to several geometric graph classes.

2.1 Partial cubes

The lattice dimension of a graph is the smallest k such that it isometrically embeds in the Cartesian product of k paths. This parameter only exists for partial cubes, *a.k.a.*, isometric subgraphs of hypercubes (where, by a hypercube, we mean a Cartesian product of edges). The isometric dimension of a partial cube is the least τ such that it isometrically embeds in the τ -dimensional hypercube.

► **Theorem 2.** *All eccentricities in an n -vertex partial cube with lattice dimension k and isometric dimension τ can be computed in $\mathcal{O}((\tau^2 + 2^{\mathcal{O}(k \log k)}) \cdot n^{1+o(1)})$ time.*

This is truly subquadratic provided $k = o(\log n)$ and $\tau = \mathcal{O}(n^{0.5-\varepsilon})$, for some $\varepsilon > 0$.

Proof. Let $G = (V, E)$ be a partial cube. It is well-known that E can be partitioned in so-called θ -classes, where each class represents a dimension of the smallest hypercube in which G can be isometrically embedded [34, 75]. Furthermore, given an edge, we can compute its θ -class in linear time (see Sec. 3 in [49]). Therefore, we can isometrically embed G in a smallest hypercube in $\tilde{\mathcal{O}}(\tau n)$ (here, we implicitly use that G only has $\tilde{\mathcal{O}}(n)$ edges, see Lemma 2 in [49]). Being given such an embedding, Eppstein's algorithm computes an embedding of G in the Cartesian product of a least number of paths in $\mathcal{O}(\tau^2 n)$ time [48]. Then, we are done applying Theorem 1 to the resulting embedding. ◀

2.2 Triangular and hexagonal systems

A *triangular system* is a subgraph of the regular triangular grid which is induced by the vertices lying on a simple circuit and inside the region bounded by this circuit. Similarly, a hexagonal system (*a.k.a.*, benzenoid) is a subgraph of the regular hexagonal grid bounded by a simple circuit. Improving on two prior works [21, 23], but at the price of a slightly slower running time, we prove that:

► **Theorem 3.** *All eccentricities in an n -vertex triangular system can be computed in $\tilde{\mathcal{O}}(n)$ time. All eccentricities in an n -vertex hexagonal system can be computed in $\tilde{\mathcal{O}}(n)$ time.*

Proof. Every hexagonal system can be isometrically embedded in the Cartesian product of three trees, in linear time [21]. Then, we are done applying Theorem 1. The same holds for triangular system, but it is a scale embedding: all distances in the tree are multiplied by two [23]. As it shall become clearer in Sec. 3.2, our framework easily accommodates to this more general type of embeddings. ◀

2.3 Polygon graphs

A circle graph is the intersection graph of chords in a cycle. For every $k \geq 2$, a k -polygon graph is the intersection graph of chords in a convex k -polygon where the ends of each chord lie on two different sides. Note that the k -polygon graphs form an increasing hierarchy of all the circle graphs. We obtain:

► **Theorem 4.** *All eccentricities in an n -vertex m -edge k -polygon graph can be computed in $\mathcal{O}(2^{\mathcal{O}(k)}(n+m)^{1+o(1)})$ time, within an additive one-sided error of at most two.*

Proof. Every k -polygon graph can be embedded in a system of $2 \log_{3/2} k + 7$ spanning trees with stretch two. Furthermore, such a system can be computed in linear time, if the graph is given together with its intersection model [36]. It was observed in [72] that an intersection model can be computed in $\mathcal{O}(4^k(n+m)\alpha(n+m))$ time, where $\alpha(\cdot, \cdot)$ denotes the Ackermann inverse function. We are now done applying Theorem 1 to the system (for $k' = \mathcal{O}(\log k)$). ◀

3 Proof of Theorem 1

We devote this section to the proof of our first main result in the paper (Theorem 1). In Sec. 3.1, we reduce to a more general query-answering problem on trees. We then solve this problem in Sec. 3.2.

3.1 Reductions

It turns out that *all* three types of embeddings that are considered in Theorem 1 can be reduced to the design of an efficient data structure for some abstract problem over a system of trees. In what follows, \odot denotes a binary associative operation over the nonnegative real numbers (*e.g.*, the addition, minimum or maximum of two numbers).

► **Problem 1** (\odot -ECCENTRICITIES).

Global Input: A system $(T_i)_{1 \leq i \leq k}$ of trees, and a subset $S \subseteq \prod_{i=1}^k V(T_i)$.

Query Input: $v = (v_1, v_2, \dots, v_k) \in \prod_{i=1}^k V(T_i)$

Query Output: $e_{\odot}(v, S) := \max\{d_{T_1}(s_1, v_1) \odot d_{T_2}(s_2, v_2) \odot \dots \odot d_{T_k}(s_k, v_k) \mid (s_1, s_2, \dots, s_k) \in S\}$.

Due to lack of space, formal reductions are omitted from the paper. Let us only sketch one of them (we stress that all these reductions are pretty similar to each other). Specifically, let φ be an isometric embedding of a graph $G = (V, E)$ (or of a discrete metric space) to the Cartesian product of the trees T_1, T_2, \dots, T_k . We set $S := \varphi(V)$, and then, for every $u \in V$ we obtain: $e_+(\varphi(u)) = \max_{v \in V} \sum_{i=1}^k d_{T_i}(\varphi_i(u), \varphi_i(v)) = \max_{v \in V} d_{\square_{i=1}^k T_i}(\varphi(u), \varphi(v))$, where \square stands for the Cartesian product and, for every $1 \leq i \leq k$, φ_i denotes the projection of φ to $V(T_i)$. In particular, $e_+(\varphi(u))$ is equal to the eccentricity of u . Similarly, we can reduce the eccentricity problem in systems and strong products of trees to min- and max-ECCENTRICITIES, respectively. We also stress that being given a *quasi* isometric embedding, our above approach leads to approximation algorithms for computing all the eccentricities.

3.2 A range-query framework

Our main result in this section is as follows:

► **Theorem 5.** For every $(T_i)_{1 \leq i \leq k}$ and S , let $N := \sum_{i=1}^k |V(T_i)|$. We can solve min-ECCENTRICITIES (resp., +-ECCENTRICITIES) with $\mathcal{O}(2^{\mathcal{O}(k \log k)}(N+|S|)^{1+o(1)})$ pre-processing time and $\mathcal{O}(2^{\mathcal{O}(k \log k)}(N+|S|)^{o(1)})$ query time.

We need to introduce two useful tools. First, let V be a set of k -dimensional points, and let $f : V \rightarrow \mathbb{R}$. A *box* is the Cartesian product of k intervals. A range query asks, given a box \mathcal{R} , for a point $\vec{p} \in V \cap \mathcal{R}$ s.t. $f(\vec{p})$ is maximized. Up to some pre-processing in $\mathcal{O}(|V| \log^{k-1} |V|)$ time, such query can be answered in $\mathcal{O}(\log^{k-1} |V|)$ time [74]. The corresponding data structure is called a k -dimensional range tree. Furthermore, note that for any $\varepsilon > 0$, $\forall x > 0$, $\log^k x \leq 2^{\mathcal{O}(k \log k)} x^\varepsilon$ [17].

Second, for an n -node tree $T = (V, E)$, a *centroid* is a node whose removal leaves subtrees of order at most $n/2$. A classic theorem from Jordan asserts that such node always exists [62]. Furthermore, we can compute a centroid in $\mathcal{O}(n)$ time by dynamic programming (e.g., see [56]). A *centroid decomposition* of T is a rooted tree T' , constructed as follows. If $|V(T)| \leq 1$, then $T' = T$. Otherwise, let c be a centroid. Let T'_1, T'_2, \dots, T'_p be centroid decompositions for the subtrees T_1, T_2, \dots, T_p of $T \setminus \{c\}$. We obtain T' from T'_1, T'_2, \dots, T'_p by adding an edge between c and the respective roots of these rooted subtrees, choosing c as the new root. Note that we can compute a centroid decomposition in $\mathcal{O}(n)$ time [32]. We rather use the folklore $\mathcal{O}(n \log n)$ -time algorithm since, for any node v , we also want to store its path $P(v)$, in T' , until the root, and the distances $d_T(v, c_i)$, in T , for any $c_i \in P(v)$.

Sketch proof of Theorem 5. Due to lack of space, we only give a proof for the case $\odot = +$. During a pre-processing phase, we compute a centroid decomposition for each tree T_i , $1 \leq i \leq k$ separately, in total $\mathcal{O}(N \log N)$ time. Then, we iterate over the elements $s = (s_1, s_2, \dots, s_k) \in S$. For every $1 \leq i \leq k$, let $P(s_i)$ be the path of s_i to the root into the centroid decomposition T'_i computed for T_i . We consider all possible k -sequences $c = (c_1, c_2, \dots, c_k)$ s.t., $\forall 1 \leq i \leq k$, $c_i \in P(s_i)$. Since the length of each path $P(s_i)$ is in $\mathcal{O}(\log N)$, there are $\mathcal{O}(\log^k N)$ possibilities. W.l.o.g., all nodes have a unique identifier. Throughout the remainder of the proof, we identify the nodes with their identifiers, thus treating them as numbers. For any sequence c , we create an $2k$ -dimensional point $\vec{p}_c(s)$, as follows: for every $1 \leq i \leq k$, the $(2i-1)^{th}$ and $2i^{th}$ coordinates are equal to c_i and the unique neighbour of c_i onto the $c_i s_i$ -path in T'_i , respectively (if $c_i = s_i$, then we may set both coordinates equal to s_i). The construction of all these $\mathcal{O}(|S| \log^k N)$ points takes time $\mathcal{O}(k|S| \log^k N)$. We include these points $\vec{p}_c(s)$, with an associated value $f(\vec{p}_c(s))$ (to be specified later in the proof) in some $2k$ -dimensional range tree. It takes $\mathcal{O}(|S| \log^k N) \log^{2k-1}(|S| \log^k N) = \mathcal{O}(2^{\mathcal{O}(k \log k)}(N+|S|)^{1+o(1)})$ time.

Then, in order to answer a query, let $v = (v_1, v_2, \dots, v_k)$ be the input. As before, for every $1 \leq i \leq k$, let $P(v_i)$ be the path of v_i to the root into T'_i . We iterate over all the k -sequences $c = (c_1, c_2, \dots, c_k)$ s.t., $\forall 1 \leq i \leq k$, $c_i \in P(v_i)$. Let $S_c \subseteq S$ contain every (s_1, s_2, \dots, s_k) s.t. $\forall 1 \leq i \leq k$, c_i is the least common ancestor of v_i and s_i in T'_i . The subsets S_c partition S , and so, $e_+(v, S) = \max_c e_+(v, S_c)$. We are left explaining how to compute $e_+(v, S_c)$ for any fixed c . For that, we observe that c_i is the least common ancestor of v_i and s_i in T'_i if and only if $c_i \in P(s_i) \cap P(v_i)$, and either $c_i \in \{v_i, s_i\}$ or the two neighbours of c_i onto the $c_i v_i$ -path and $c_i s_i$ -path in T'_i are different. In the latter case, let us denote by u_i the neighbour of c_i onto the $c_i v_i$ -path in T'_i . We can check these above conditions, for every $1 \leq i \leq k$, with the following constraints, over the $2k$ -dimensional points $\vec{p} = (p_1, p_2, \dots, p_{2k})$ constructed during the pre-processing phase: $\forall 1 \leq i \leq k$, $p_{2i-1} = c_i$ and if $c_i \neq v_i$, $p_{2i} \neq u_i$. Since $p_{2i} \neq u_i$ is equivalent to $p_{2i} \in (-\infty, u_i) \cup (u_i, +\infty)$, each inequality can be replaced by two range constraints over the same coordinate. In

particular, since there are $\leq k$ such inequalities, we can transform these above constraints into $\mathcal{O}(2^k)$ range queries. Therefore, we can output a point $\vec{p}_c(s)$, for some $s \in S_c$, maximizing $f(\vec{p}_c(s))$, in $\mathcal{O}(2^k \log^{2k-1}(|S| \log^k N)) = \mathcal{O}(2^{\mathcal{O}(k \log k)}(N + |S|)^{o(1)})$ time. Let $f(\vec{p}_c(s)) = \sum_{i=1}^k d_{T_i}(s_i, c_i)$. Since we have $s \in S_c$, $d_{T_i}(s_i, v_i) = d_{T_i}(s_i, c_i) + d_{T_i}(c_i, v_i)$ [54]. As a result: $e_+(v, S_c) = f(\vec{p}_c(s)) + \sum_{i=1}^k d_{T_i}(c_i, v_i)$. There are $\mathcal{O}(\log^k N)$ possible c , and so, the final query time is in $\mathcal{O}(2^{\mathcal{O}(k \log k)}(N + |S|)^{o(1)})$.

For the case when $\odot = \min$, we need points with $\mathcal{O}(k)$ more coordinates in order to correctly identify some index i s.t. $d_{T_i}(v_i, s_i) = \min_{1 \leq j \leq k} d_{T_j}(v_j, s_j)$. This is a similar trick as the one used in [1, 17, 19]. \blacktriangleleft

In contrast to Theorem 5, the distance between two vertices in the strong product of k trees equals the maximum distance between their k respective projections. Hence, we can process each tree of the system separately, and we obtain:

► **Lemma 6.** *For every $(T_i)_{1 \leq i \leq k}$ and S , let $N := \sum_{i=1}^k |V(T_i)|$. We can solve max-ECCENTRICITIES with $\mathcal{O}(N + k|S|)$ pre-processing time and $\mathcal{O}(k)$ query time.*

Proof. For every $1 \leq i \leq k$, let $\varphi_i : S \rightarrow V(T_i)$ be the projection of S to T_i . We stress that the projections $\varphi_i(S)$, $1 \leq i \leq k$, can be computed in total $\mathcal{O}(k|S|)$ time. Then, we iteratively remove from T_i the leaves that are not in $\varphi_i(S)$. Let T'_1, T'_2, \dots, T'_k be the k subtrees resulting from this above pre-processing. Note that, for every $1 \leq i \leq k$, $T_i \setminus T'_i$ is a forest whose each subtree can be rooted at some node adjacent to a leaf of T'_i , and so, adjacent to a node of $\varphi_i(S)$. For every node $v_i \in V(T_i) \setminus V(T'_i)$, let $\phi(v_i)$ be the unique leaf of T'_i s.t. the subtree of $T_i \setminus T'_i$ that contains v_i also contains a neighbour of $\phi(v_i)$. We compute, and store, the distance $d_{T_i}(v_i, \phi(v_i))$. Since, for doing so, we only need to perform breadth-first searches on disjoint subtrees, the total running time of this step is in $\mathcal{O}(N)$. Finally, we compute, for $1 \leq i \leq k$, all eccentricities in T'_i . Again, this can be done in total $\mathcal{O}(N)$ time (e.g., see [22, 43]). This concludes the pre-processing phase.

In order to answer a query, let us consider some input $v = (v_1, v_2, \dots, v_k)$. Our key insight here is that we have:

$$e_{\max}(v, S) = \max_{1 \leq i \leq k} \{ \max_{(s_1, s_2, \dots, s_k) \in S} d_{T_i}(v_i, s_i) \} = \max_{1 \leq i \leq k} \{ \max_{s_i \in \varphi_i(S)} d_{T_i}(v_i, s_i) \}.$$

Therefore, in order to compute $e_{\max}(v, S)$ in $\mathcal{O}(k)$ time, it suffices to compute $\max\{d_{T_i}(v_i, s_i) \mid s_i \in \varphi_i(S)\}$ in $\mathcal{O}(1)$ time for every $1 \leq i \leq k$. If $v_i \in V(T'_i)$ then, since $\varphi_i(S) \subseteq V(T'_i)$ and furthermore all the leaves of T'_i are in $\varphi_i(S)$, we get $\max\{d_{T_i}(v_i, s_i) \mid s_i \in \varphi_i(S)\} = e_{T'_i}(v_i)$. Otherwise, $\max\{d_{T_i}(v_i, s_i) \mid s_i \in \varphi_i(S)\} = d_{T_i}(v_i, \phi(v_i)) + e_{T'_i}(\phi(v_i))$. \blacktriangleleft

Theorem 1 now follows from our reductions in Sec. 3.1 combined with Theorem 5 and Lemma 6.

4 Hardness results

We complete the positive results of Theorem 1 with two conditional lower bounds. Both theorems follow from a “SETH-hardness” result in order to compute the diameter of *split graphs* with a logarithmic³ clique-number [13], and from the observation that every split graph with a maximal clique K embeds in any system of $|K|$ shortest-path trees rooted at the vertices of K .

³ The logarithmic upper bound is not explicitly stated in [13], but it easily follows from the sparsification lemma applied to k -SAT.

► **Theorem 7.** *For any $\varepsilon > 0$, there exists a $c(\varepsilon)$ s.t., under SETH, we cannot compute the diameter of n -vertex graphs in $\mathcal{O}(n^{2-\varepsilon})$ time, even if we are given as input an isometric embedding of the graph in a system of at most $c(\varepsilon) \log n$ spanning trees. In particular, under SETH, there is no data structure for min-ECCENTRICITIES with $\mathcal{O}(2^{o(k)}(N + |S|)^{1-o(1)})$ pre-processing time and $\mathcal{O}(2^{o(k)}(N + |S|)^{o(1)})$ query time, where $N := \sum_{i=1}^k |V(T_i)|$.*

We believe our Theorem 7 to be important since the embeddings of graphs in systems of tree spanners are well-studied in the literature [35, 36, 37, 38, 39, 40, 41].

► **Theorem 8.** *For any $\varepsilon > 0$, there exists a $c(\varepsilon)$ s.t., under SETH, we cannot compute the diameter of n -point metric spaces in $\mathcal{O}(n^{2-\varepsilon})$ time, even if we are given as input an isometric embedding of the space in a Cartesian product of at most $c(\varepsilon) \log n$ tree factors. In particular, under SETH, there is no data structure for +-ECCENTRICITIES with $\mathcal{O}(2^{o(k)}(N + |S|)^{1-o(1)})$ pre-processing time and $\mathcal{O}(2^{o(k)}(N + |S|)^{o(1)})$ query time, where $N := \sum_{i=1}^k |V(T_i)|$.*

Proof. Recall that for any $\varepsilon > 0$, there exists a $c(\varepsilon)$ s.t., under SETH, we cannot compute the diameter in $\mathcal{O}(n^{2-\varepsilon})$ time on the split graphs of order n and clique-number at most $c(\varepsilon) \log n$ [13]. Let $G = (K \cup I, E)$ be a split graph, where K and I are a clique and a stable set, respectively. If it is not given, such a bipartition of $V(G)$ can be computed in linear time [57]. For every $u \in K$, we construct a tree T'_u , and an embedding φ_u of G into the latter, as follows. We start from a single-node tree, to which we map vertex u . Then, for every $v \in N_G(u)$, we add a leaf into the tree, adjacent to the image of u , to which we map the vertex v . We add another node u^* in T'_u , that is also adjacent to the image of u (note that u^* is *not* the image of a vertex of G). Finally, for every vertex $v \in V(G) \setminus N_G[u]$, we add a leaf node, to which we map vertex v , that we connect to u^* by a path of length two. Let $\varphi : V(G) \rightarrow V(\square_{u \in K} T'_u)$ be s.t., for every $v \in V(G)$, $\varphi(v) = (\varphi_u(v))_{u \in K}$. The metric space considered is $(\varphi(V(G)), d)$, where d is the sub-metric induced by $d_{\square_{u \in K} T'_u}$ (distances in the Cartesian product). Indeed, let $v, v' \in V(G)$. For every $u \in K$, by construction we have $\text{diam}(T'_u) = 4$, and so, $d_{T'_u}(\varphi_u(v), \varphi_u(v')) \leq 4$. Specifically, if $u \in \{v, v'\}$ then $d_{T'_u}(\varphi_u(v), \varphi_u(v')) \leq 3$; if $v, v' \in N_G(u)$ then $d_{T'_u}(\varphi_u(v), \varphi_u(v')) = 2$; otherwise, $d_{T'_u}(\varphi_u(v), \varphi_u(v')) = 4$. Altogether combined, if $d_G(v, v') = 3$ (in particular, $v, v' \in I$) then we get $d(\varphi(v), \varphi(v')) = 4|K|$, otherwise we get $d(\varphi(v), \varphi(v')) \leq 3 + 4(|K| - 1) = 4|K| - 1$. ◀

5 One-to-many tree embeddings

We end up discussing a different type of tree-like embedding than in Sec. 2. First we present an algorithm for computing all eccentricities being given such an embedding (Sec. 5.1). We then propose, in Sec 5.2, an application of our result to the absolute center problem in a subclass of median graphs.

5.1 Computation of subset-eccentricities

We now address the problem of computing the eccentricity of *subsets* of nodes, in *one* tree:

► **Theorem 9.** *Let T be any n -node tree, and let $\alpha : V(T) \rightarrow \mathbb{R}$. After a pre-processing in $\mathcal{O}(n \log n)$ time, for any subset U of nodes and function $\beta : U \rightarrow \mathbb{R}$, we can compute the value $e_{T,\alpha}(U, \beta) = \max_{v \in V(T)} \min_{u \in U} (\alpha(v) + d_T(v, u) + \beta(u))$ in $\mathcal{O}(|U| \log^2 n)$ time.*

In particular, after an $\mathcal{O}(n \log n)$ -time pre-processing we can compute $e_T(U)$ in $\mathcal{O}(|U| \log^2 n)$ time by setting $\alpha(v) = 0$ for every node $v \in V(T)$ and $\beta(u) = 0$ for every node $u \in U$. The proof of Theorem 9 is technical, and we omit it due to lack of space. It is based

on a completely different processing method of the tree than in Sec. 3.2: using heavy-path decomposition [71] and the local computation of so-called Cartesian trees for maximum range queries [53].

Here is a possible application of our Theorem 9 for general graphs. By a clan embedding of a graph $G = (V, E)$ in a tree T , we mean a one-to-many embedding $S : V \rightarrow 2^{V(T)}$ such that, to each vertex $v \in V$, we also associate a leader $\chi(v) \in S_v$ in its corresponding node-subset of T . This embedding must further satisfy $d_T(S_u, S_v) \geq d_G(u, v)$ for every $u, v \in V$. The distortion of a clan embedding can be defined as $t := \max_{u \neq v} d_T(\chi(u), S_v) / d_G(u, v)$.

For each node x of T , we set $\alpha(x)$ to 0 if $x = \chi(u)$ for some vertex $u \in V$; otherwise, we set $\alpha(x)$ to some large enough *negative* value. Then, for any subset $U \subseteq V$, we can apply Theorem 9 to $\bigcup_{u \in U} S_u$ in order to compute a t -approximation of $e_G(U)$. We refer to [52] for various trade-offs between the size of the subsets S_u and the resulting distortion t .

5.2 Application to a distance-labelling scheme

We devote our last section to cube-free median graphs. Recall that a graph $G = (V, E)$ is called *median* if, for any triple $x, y, z \in V$, there exists a unique vertex c that is simultaneously on some shortest xy -, yz - and zx -paths. This class is ubiquitous in Theoretical Computer Science. Indeed, the median graphs are exactly the 1-skeletons of CAT(0) cube complexes [58], the domains of event structures [67] and the solution sets of 2-SAT formulas [66], among many characterizations. The *dimension* of a median graph G is the largest $d \geq 1$ such that G contains a d -cube (hypercube of dimension d) as an induced subgraph. In particular, the median graphs of dimension 1 are exactly the trees. The median graphs of dimension at most 2, *a.k.a.*, *cube-free median graphs*, have already received some attention in the literature [7, 16, 25, 26, 27].

In what follows, we abusively call eccentricity of a point $x \in X_G$, denoted by $e_G(x)$, its maximum distance to a vertex of G .

► **Theorem 10.** *There is an $\tilde{O}(n)$ -time algorithm for computing the absolute center of n -vertex cube-free median graphs. More generally, the algorithm encodes in $\mathcal{O}(n)$ space the eccentricity of all the points of X_G .*

The remainder of this section is devoted to the proof of Theorem 10. Given any point $x \in X_G$, if $W \subseteq V$ then, let $e_G(x, W) = \max_{w \in W} d(x, w)$. We observe that:

► **Lemma 11.** *For a bipartite graph $G = (V, E)$ and an edge $uv \in E$, let $W_{u,v} = \{w \in V \mid d(u, w) < d(v, w)\}$. If x is a point of the edge uv such that $d(u, x) = t$, then, we have $e(x) = \max\{t + e(u, W_{u,v}), 1 - t + e(v, W_{v,u})\}$.*

Proof. We have $V = W_{u,v} \cup W_{v,u}$ because G is bipartite. Let $w \in W_{u,v}$. Every xw -path going by vertex v would have length $1 - t + d(v, w) = 2 - t + d(u, w) > t + d(u, w)$, and therefore, it cannot be a shortest xw -path. In the same way, let $w \in W_{v,u}$. Every xw -path going by vertex u would have length $t + d(u, w) = 1 + t + d(v, w) > 1 - t + d(v, w)$, and therefore, it cannot be a shortest xw -path either. ◀

As a result of Lemma 11, we are left computing the values $e(u, W_{u,v})$ and $e(v, W_{v,u})$ for every edge uv . Since the cube-free median graphs are sparse [26], there are only $\mathcal{O}(n)$ values to be stored. Furthermore, for median graphs, the subsets $W_{u,v}$ are called half-spaces and they induce convex subgraphs. Thus, in a way, we could reduce the absolute center problem to the computation of all eccentricities in various convex subgraphs of the input. However, the number of subgraphs to be considered is in general super-constant, even for the cube-free

median graphs. Since by Lemma 11, a point in the absolute center must be either a vertex or at the middle of an edge, we could also reduce the absolute center problem to computing all eccentricities in the subdivision of G . Unfortunately, median graphs are not closed by taking subdivisions. We take an alternative approach in what follows.

For that, we first recall some notions and results from [26]. These results are specific to cube-free median graphs but, as we pointed it out in Sec. 1, similar structural decomposition theorems were proved in [24, 29, 28] for completely unrelated geometric graph classes.

In what follows, let $G = (V, E)$ be a cube-free median graph. A *centroid* is any vertex minimizing the sum of its distances to all other vertices. Recently, it was shown that a centroid in a median graph can be computed in linear time [10]. So, let $c \in V$ be a centroid.

A subgraph H of G is *gated* if, for every $v \in V \setminus V(H)$, there exists a $v^* \in V(H)$ s.t., $\forall u \in V(H)$, $d_G(u, v) = d_G(u, v^*) + d_G(v^*, v)$. We define the *fibers* $F(x) = \{x\} \cup \{v \in V(G \setminus H) \mid x \text{ is the gate of } v \text{ in } H\}$. The fibers $F(x)$, $x \in V(H)$ partition the vertex-set of G , and each induces a gated subgraph [26].

For any $z \in V$, the *star* $St(z)$ of z is the subgraph of G induced by all edges and squares of G incident to z . Any such star $St(z)$ is gated and, if furthermore $z = c$, every fiber $F(x)$, $x \in St(c)$ contains at most $|V|/2$ vertices [26].

A fiber $F(x)$ of the star $St(c)$ is a *panel* if $x \in N_G(c)$, and a *cone* otherwise. We say that two fibers $F(x), F(y)$ are *neighboring* if there exists an edge with an end in $F(x)$ and the other end in $F(y)$. If two fibers are neighboring then one must be a panel and the other must be a cone; furthermore, a cone has two neighboring panels [26]. Two fibers are *2-neighboring* if they are cones adjacent to the same panel. Finally, two fibers that are neither neighboring nor 2-neighboring are called *separated*.

The subset of vertices in $F(x)$ with a neighbour in $F(y)$ is denoted by $\partial_y F(x)$ (with the understanding that $\partial_y F(x) = \emptyset$ when $F(x), F(y)$ are not neighboring). The *total boundary* of $F(x)$ is defined as $\partial^* F(x) = \cup_y \partial_y F(x)$.

For a set of vertices A , an *imprint* of a vertex u is a vertex $a \in A$ such that there is no vertex of A (but a itself) on any shortest au -path. A subgraph H of G is *quasigated* if every vertex of $V(G \setminus H)$ has at most two imprints. It is known that for each fiber $F(x)$ of a star $St(z)$, the total boundary $\partial^* F(x)$ is an isometric quasigated tree [26].

► **Lemma 12** ([26]). *Let $G = (V, E)$ be a cube-free median graph, let $c \in V$ be a centroid, and let $F(x), F(y)$ be two fibers of the star $St(c)$. The following hold for every $u \in F(x)$, $v \in F(y)$.*

- *If $F(x)$ and $F(y)$ are separated, then $d_G(u, v) = d_G(u, c) + d_G(c, v)$;*
- *If $F(x)$ and $F(y)$ are neighboring, $F(x)$ is a panel and $F(y)$ is a cone, then let u_1, u_2 be the two (possibly equal) imprints of u on the total boundary $\partial^* F(x)$, and let v^* be the gate of v in $F(x)$. We have $d_G(u, v) = \min\{d_G(u, u_1) + d_G(u_1, v^*) + d_G(v^*, v), d_G(u, u_2) + d_G(u_2, v^*) + d_G(v^*, v)\}$;*
- *If $F(x)$ and $F(y)$ are 2-neighboring, then let $F(w)$ be the panel neighboring $F(x)$ and $F(y)$. Let u^* and v^* be the gates of u and v in $F(w)$. Then $d_G(u, v) = d_G(u, u^*) + d_G(u^*, v^*) + d_G(v^*, v)$.*

Sketch proof of Theorem 10. We compute for each edge $uv \in E$ two values, denoted by $f_G(u, v)$ and $f_G(v, u)$, so that: $f_G(u, v) \leq e_G(u, W_{u,v})$, $f_G(v, u) \leq e_G(v, W_{v,u})$ and, for each point x of the edge such that $d(u, x) = t$, $e_G(x) = \max\{t + f_G(u, v), 1 - t + f_G(v, u)\}$. If $E = \{uv\}$ then, we set $f_G(u, v) = f_G(v, u) = 0$. Otherwise, we compute a centroid c and we use an algorithmic procedure from [26] in order to compute in $\mathcal{O}(n)$ time the vertex-set and the edge-set of all fibers $F(x)$, $x \in St(c)$. We enumerate all the fibers $F(x)$, $x \in St(c)$, and we compute $D(x) = \max_{v \in F(x)} d_G(v, c)$. It can be done in total $\mathcal{O}(n)$ time by performing a

BFS rooted at c . Also, for each $x \in St(c)$, let G_x be induced by $F(x)$. Since, $F(x)$ is gated, and so convex, G_x is a cube-free median graph. In particular, we can apply our algorithm recursively on it in order to compute the values $f_{G_x}(\cdot, \cdot)$ associated to its edges. Then, we apply the following steps (some of which not being detailed due to lack of space):

Step 1. We consider all edges with their both ends in $St(c)$ and we compute the corresponding values $f_G(\cdot, \cdot)$. Roughly, it can be done in total $\mathcal{O}(n \log n)$ time if we order the vertices of $St(c)$ by non-decreasing $D(\cdot)$ value, and if we keep track for each $x \in St(c)$ of its neighbours in the star. To illustrate this, consider for example some edge xc . Then, $W_{x,c} = \bigcup \{F(y) \mid y \in N[x] \cap (St(c) \setminus \{c\})\}$, and so it becomes possible to compute $f_G(x, c), f_G(c, x)$ from the $D(\cdot)$ values. In the same way for an edge xy , where $x \in N(c), y \in St(c) \setminus N[c]$, we have $W_{y,x} = F(y) \cup F(x')$ where $F(x), F(x')$ are the two panels neighboring $F(y)$, and so, we can proceed as in the previous case for computing $f_G(x, y), f_G(y, x)$. Thus from now on, we only consider edges with at least one end not in $St(c)$.

Step 2. We consider each remaining edge uv sequentially, where $d(u, c) < d(v, c)$. Let $x, y \in St(c)$ satisfy $u \in F(x), v \in F(y)$. Let A be the union of all the fibers $F(z)$ s.t. $F(z)$ is separated from both $F(x)$ and $F(y)$. By Lemma 12, we have $A \subseteq W_{u,v}$. We want to compute $\max_{w \in A} d_G(u, w)$. There are three cases: $x = y$ and $F(x)$ is a panel; $x = y$ and $F(x)$ is a cone; $x \neq y$. Due to lack of space, we only detail the first case (the other two cases can be handled with similar techniques). Specifically, let $z \neq x$ be such that $F(z)$ is a panel and $D(z)$ is maximized. It can be computed in constant-time assuming the panels were ordered during a pre-processing phase. Recall that two panels are always separated. Therefore, the maximum distance between u and a vertex of A in a panel is equal to $d(u, c) + D(z)$. The case of separated cones is more complicated, and it requires some pre-processing. Specifically, we assign to all the fibers $F(z)$ pairwise different positive numbers, that we abusively identify with the vertices z of the star $St(c)$. Then, we enumerate the cones $F(z')$, $z' \in St(c) \setminus N_G[c]$. Let $F(z_1), F(z_2)$ be the two panels neighboring $F(z')$. We create a point $\vec{p}(z') = (z_1, z_2)$, to which we associate the value $f(\vec{p}(z')) = D(z')$. To complete the pre-processing, we put these points and their associated values in a 2-dimensional range tree, that takes $\mathcal{O}(n \log n)$ time. Now, to compute the maximum value between u and a vertex of A in a cone, it suffices to compute a point $\vec{p}(z') = (p_1, p_2)$ such that:

$$\begin{cases} p_1 \neq x, p_2 \neq x \\ f(\vec{p}(z')) \text{ is maximized for these above properties.} \end{cases}$$

Indeed, this maximum distance is exactly $d_G(u, c) + D(z') = d_G(u, c) + f(\vec{p}(z'))$. Furthermore, since each inequality gives rise to two disjoint intervals to which the corresponding coordinate must belong, a point $\vec{p}(z')$ as above can be computed using four range queries. It takes $\mathcal{O}(\log n)$ time. As a result, the total running time of this step is in $\mathcal{O}(n \log n)$.

Step 3. We consider each edge uv with at least one end not in $St(c)$ and such that $u \in F(x)$ for some panel $F(x)$. In what follows, for any $z, z' \in St(c)$, we write $z \sim z'$ if $F(z), F(z')$ are neighboring. Let $B = \bigcup \{F(y) \mid x \sim y \text{ and } v \notin F(y)\}$. Intuitively, what we try to do at this step is to compute $e(u, W_{u,v} \cap B)$ and $e(v, W_{v,u} \cap B)$. There are two cases: either $v \notin F(x)$ or $v \in F(x)$. In both cases, we reduce our computations to some suitable eccentricity problem on the tree $T = \partial^* F(x)$. Next, we detail the case $v \in F(x)$, which is the one to which we need to apply Theorem 9 (the case $v \notin F(x)$ is solved by using a rather standard dynamic

programming approach on the tree T). For each node $z \in V(T)$, let $\alpha(z)$ be the maximum distance between z and any vertex w in a neighboring cone $F(y)$ of which z is the gate in $F(x)$ (with the understanding that, if no vertex has z as its gate, then $\alpha(z)$ is a sufficiently large *negative* value, say $\alpha(z) = -|V(T)|$). All these nodes weights can be pre-computed in $\mathcal{O}(\sum_{y|x \sim y} |F(y)|)$ time. Furthermore, note that a cone is neighboring two panels [26]. As a result, if we consider each panel $F(x)$ sequentially then, we scan each cone only twice, and the total running-time of this pre-computation phase is in $\mathcal{O}(n)$.

We compute $e_G(u, B)$, $e_G(v, B)$ and $e_G(\{u, v\}, B) =^{def} \max_{w \in B} d_G(w, \{u, v\})$. For that, let $u_1, u_2 \in V(T)$ be the two (possibly equal) imprints of u , and similarly let $v_1, v_2 \in V(T)$ be the imprints of v . Set $\beta(u_1) = d_G(u, u_1)$, $\beta(u_2) = d_G(u, u_2)$ and in the same way $\beta(v_1) = d_G(v, v_1)$, $\beta(v_2) = d_G(v, v_2)$. Since T is isometric, then it follows from the distance formulae in Lemma 12 that the values to be computed are exactly $e_{T,\alpha}(\{u_1, u_2\}, \beta)$, $e_{T,\alpha}(\{v_1, v_2\}, \beta)$ and $e_{T,\alpha}(\{u_1, v_1, u_2, v_2\}, \beta)$. By Theorem 9, the latter can be computed in $\mathcal{O}(\log^2 n)$ time, up to some initial pre-processing of T in $\mathcal{O}(|V(T)| \log |V(T)|) = \mathcal{O}(|F(x)| \log |F(x)|)$ time. Overall, the running-time of this step is in $\mathcal{O}(n \log^2 n)$.

Let $e_G(u, B) = p_1$, $e_G(v, B) = p_2$ and $e_G(\{u, v\}, B) = p$. W.l.o.g., $p_1 \leq p_2$. It implies $p_2 = p + 1$. Then, $e_G(u, B \cap W_{u,v}) = p$. In the same way, if $p_1 = p + 1$ then we also have $e_G(v, B \cap W_{v,u}) = p$. From now on, we assume $p_1 < p + 1$. In particular, $p_1 = p$. But then, $e_G(v, B \cap W_{v,u}) \leq p - 1$, and therefore we needn't compute this value (*i.e.*, because we always have $t + p \geq 1 - t + e_G(v, B \cap W_{v,u})$ for any $t \in (0; 1)$).

Step 4. Finally, consider each edge uv with at least one end not in $St(c)$ and such that $v \in F(y)$ for some cone $F(y)$. Let $C = \bigcup \{F(y') \mid F(y) \text{ and } F(y') \text{ are either neighboring or 2-neighbouring, and } u \notin F(y')\}$. We would like to compute $e_G(u, W_{u,v} \cap C)$ and $e_G(v, W_{v,u} \cap C)$. There are two cases: either $u \in F(y)$ or $u \notin F(y)$. Consider the case $u \in F(y)$ (the other case can be dealt with similarly). We consider each x s.t. $F(x)$ is a panel neighboring $F(y)$ sequentially (there are only two such x). Let C_x contain $F(x)$ and all cones of C that are neighboring $F(x)$. We now consider $T = \partial^* F(x)$ which we assume to be pre-processed as during the previous Step 3. Let $u^*, v^* \in V(T)$ be the respective gates of u, v . Indeed, by Lemma 12 there is always a shortest-path between u (resp., v) and any vertex of C_x that goes by u^* (resp., by v^*). This part is solved through a delicate case analysis which depends on: some values $f_{G_x}(\cdot, \cdot)$ (obtained by applying our algorithm recursively to G_x) and some eccentricity functions computed for T during Step 3.

Overall, each fiber contains at most $n/2$ vertices, and so there are $\mathcal{O}(\log n)$ recursive stages. Since a stage runs in $\mathcal{O}(n \log^2 n)$ time (the bottleneck being Step 3), the total running time for computing all the values $f_G(\cdot, \cdot)$ is in $\mathcal{O}(n \log^3 n)$. ◀

Open problem. To which other graph classes can our framework in this section be applied? A good candidate could be the planar graphs of non positive combinatorial curvature [24], and especially the trigraphs [29]. To our best knowledge, it is open whether the eccentricities in a trigraph can be computed in almost linear time (there exists a linear-time algorithm for computing the diameter and the center [53]).

References

- 1 A. Abboud, V. Vassilevska Williams, and J. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the twenty-seventh annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–391. SIAM, 2016.
- 2 Muad Abu-Ata and Feodor F. Dragan. Metric tree-like structures in real-world networks: an empirical study. *Networks*, 67(1):49–68, 2016. doi:10.1002/net.21631.

- 3 H. Bandelt and V. Chepoi. Metric graph theory and geometry: a survey. *Contemporary Mathematics*, 453:49–86, 2008.
- 4 H. Bandelt and H. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.
- 5 H.-J. Bandelt. Recognition of tree metrics. *SIAM Journal on Discrete Mathematics*, 3(1):1–6, 1990.
- 6 H.-J. Bandelt, V. Chepoi, and D. Eppstein. Combinatorics and geometry of finite and infinite squaregraphs. *SIAM Journal on Discrete Mathematics*, 24(4):1399–1440, 2010.
- 7 H.-J. Bandelt, V. Chepoi, and D. Eppstein. Ramified rectilinear polygons: coordinatization by dendrons. *Discrete & Computational Geometry*, 54(4):771–797, 2015.
- 8 H.-J. Bandelt and M. van De Vel. Embedding topological median algebras in products of dendrons. *Proceedings of the London Mathematical Society*, 3(3):439–453, 1989.
- 9 V. Batagelj, T. Pisanski, and J. Simoes-Pereira. An algorithm for tree-realizability of distance matrices. *International Journal of Computer Mathematics*, 34(3-4):171–176, 1990.
- 10 L. Bénéteau, J. Chalopin, V. Chepoi, and Y. Vaxès. Medians in median graphs and their cube complexes in linear time. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 11 P. Bergé and M. Habib. Diameter in linear time for constant-dimension median graphs. In *XI Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS 2021)*, 2021. To appear.
- 12 J. A. Bondy and U. S. R. Murty. *Graph theory*. Springer, 2008.
- 13 M. Borassi, P. Crescenzi, and M. Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, 2016.
- 14 M. Borassi, P. Crescenzi, and L. Trevisan. An axiomatic and an average-case analysis of algorithms and heuristics for metric properties of graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 920–939. SIAM, 2017.
- 15 A. Brandstädt, V. Chepoi, and F. Dragan. The algorithmic use of hypertree structure and maximum neighbourhood orderings. *Discrete Applied Mathematics*, 82(1-3):43–77, 1998.
- 16 B. Bresar, S. Klavzar, and R. Skrekovski. On cube-free median graphs. *Discrete Mathematics*, 307(3):345–351, 2007.
- 17 K. Bringmann, T. Husfeldt, and M. Magnusson. Multivariate Analysis of Orthogonal Range Searching and Graph Distances. *Algorithmica*, pages 1–24, 2020.
- 18 S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Transactions on Algorithms (TALG)*, 15(2):1–38, 2018.
- 19 S. Cabello and C. Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Computational Geometry*, 42(9):815–824, 2009.
- 20 C. Cheng. A poset-based approach to embedding median graphs in hypercubes and lattices. *Order*, 29(1):147–163, 2012.
- 21 V. Chepoi. On distances in benzenoid systems. *Journal of chemical information and computer sciences*, 36(6):1169–1172, 1996.
- 22 V. Chepoi, F. Dragan, M. Habib, Y. Vaxès, and H. Alrasheed. Fast approximation of eccentricities and distances in hyperbolic graphs. *Journal of Graph Algorithms and Applications*, 23(2):393–433, 2019.
- 23 V. Chepoi, F. Dragan, and Y. Vaxès. Center and diameter problems in plane triangulations and quadrangulations. In *Symposium on Discrete Algorithms (SODA’02)*, pages 346–355, 2002.
- 24 V. Chepoi, F. Dragan, and Y. Vaxès. Distance and routing problems in plane graphs of non-positive curvature. *J. Algorithms*, 61:1–30, 2006.
- 25 V. Chepoi and M. Hagen. On embeddings of CAT(0) cube complexes into products of trees via colouring their hyperplanes. *Journal of Combinatorial Theory, Series B*, 103(4):428–467, 2013.
- 26 V. Chepoi, A. Labourel, and S. Ratel. Distance and routing labeling schemes for cube-free median graphs. *Algorithmica*, pages 1–45, 2020.

- 27 V. Chepoi and D. Maftuleac. Shortest path problem in rectangular complexes of global nonpositive curvature. *Computational Geometry*, 46(1):51–64, 2013.
- 28 Victor Chepoi, Arnaud Labourel, and Sébastien Ratel. Distance labeling schemes for K_4 -free bridged graphs. In *International Colloquium on Structural Information and Communication Complexity*, pages 310–327. Springer, 2020.
- 29 Victor Chepoi, Y Vaxes, and FR Dragan. Distance-based location update and routing in irregular cellular networks. In *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pages 380–387. IEEE, 2005.
- 30 D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. *ACM Transactions on Algorithms (TALG)*, 15(3):1–57, 2019.
- 31 P. Damaschke. Computing giant graph diameters. In *International Workshop on Combinatorial Algorithms (IWOCA)*, pages 373–384. Springer, 2016.
- 32 D. Della Giustina, Ni. Prezza, and R. Venturini. A new linear-time algorithm for centroid decomposition. In *String Processing and Information Retrieval*, pages 274–282. Springer International Publishing, 2019.
- 33 R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- 34 D Ž Djoković. Distance-preserving subgraphs of hypercubes. *Journal of Combinatorial Theory, Series B*, 14(3):263–267, 1973.
- 35 F. Dragan and M. Abu-Ata. Collective additive tree spanners of bounded tree-breadth graphs with generalizations and consequences. *Theoretical Computer Science*, 547:1–17, 2014.
- 36 F. Dragan, D. Corneil, E. Köhler, and Y. Xiang. Collective additive tree spanners for circle graphs and polygonal graphs. *Discrete Applied Mathematics*, 160(12):1717–1729, 2012.
- 37 F. Dragan, Y. Xiang, and C. Yan. Collective Tree Spanners for Unit Disk Graphs with Applications. *Electronic Notes in Discrete Mathematics*, 32:117–124, 2009.
- 38 F. Dragan and C. Yan. Collective tree spanners in graphs with bounded parameters. *Algorithmica*, 57(1):22–43, 2010.
- 39 F. Dragan, C. Yan, and D. Corneil. Collective Tree Spanners and Routing in AT-free Related Graphs. *Journal of Graph Algorithms and Applications*, 10(2):97–122, 2006.
- 40 F. Dragan, C. Yan, and I. Lomonosov. Collective tree spanners of graphs. *SIAM Journal on Discrete Mathematics*, 20(1):240–260, 2006.
- 41 F. Dragan, C. Yan, and Y. Xiang. Collective additive tree spanners of homogeneously orderable graphs. In *Latin American Symposium on Theoretical Informatics*, pages 555–567. Springer, 2008.
- 42 G. Ducoffe. A New Application of Orthogonal Range Searching for Computing Giant Graph Diameters. In *Symposium on Simplicity in Algorithms (SOSA)*, 2019.
- 43 G. Ducoffe. Easy computation of eccentricity approximating trees. *Discrete Applied Mathematics*, 260:267–271, 2019.
- 44 G. Ducoffe. Optimal diameter computation within bounded clique-width graphs. Technical report, arXiv, 2020. [arXiv:2011.08448](https://arxiv.org/abs/2011.08448).
- 45 G. Ducoffe and F.F. Dragan. A story of diameter, radius and (almost) helly property. *Networks*, 2021. To appear.
- 46 G. Ducoffe, M. Habib, and L. Viennot. Fast diameter computation within split graphs. In *International Conference on Combinatorial Optimization and Applications*, pages 155–167. Springer, 2019.
- 47 G. Ducoffe, M. Habib, and L. Viennot. Diameter computation on H -minor free graphs and graphs of bounded (distance) VC-dimension. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1905–1922. SIAM, 2020.
- 48 D. Eppstein. The lattice dimension of a graph. *European Journal of Combinatorics*, 26(5):585–592, 2005.
- 49 D. Eppstein. Recognizing Partial Cubes in Quadratic Time. *Journal of Graph Algorithms and Applications*, 5(2):269–293, 2011.

- 50 S. Even and A. Litman. Layered cross product—A technique to construct interconnection networks. In *Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*, pages 60–69, 1992.
- 51 A. Farley and A. Proskurowski. Computation of the center and diameter of outerplanar graphs. *Discrete Applied Mathematics*, 2(3):185–191, 1980.
- 52 A. Filtser and H. Le. Clan embeddings into trees, and low treewidth graphs. In *53rd Annual ACM Symposium on Theory of Computing (STOC 2021)*. ACM, 2021. To appear.
- 53 H. Gabow, J. Bentley, and R. Tarjan. Scaling and related techniques for geometry problems. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 135–143, 1984.
- 54 C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.
- 55 P. Gawrychowski, H. Kaplan, S. Mozes, M. Sharir, and O. Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. In *Symposium on Discrete Algorithms (SODA)*, pages 495–514. SIAM, 2018.
- 56 A. Goldman. Optimal center location in simple networks. *Transportation science*, 5(2):212–221, 1971.
- 57 M. Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.
- 58 Mikhaïl Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987. doi:10.1007/978-1-4613-9586-7_3.
- 59 S.L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations research*, 12(3):450–459, 1964.
- 60 G.Y. Handler. Minimax location of a facility in an undirected tree graph. *Transportation Science*, 7(3):287–293, 1973.
- 61 E. Howorka. On metric properties of certain clique graphs. *Journal of Combinatorial Theory, Series B*, 27(1):67–74, 1979.
- 62 C. Jordan. Sur les assemblages de lignes. *J. Reine Angew. Math*, 70(185):81, 1869.
- 63 O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems. I: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538, 1979.
- 64 S. Khuller and Y. Sussmann. The capacitated k-center problem. *SIAM Journal on Discrete Mathematics*, 13(3):403–418, 2000.
- 65 Y.-F. Lan, Y.-L. Wang, and H. Suzuki. A linear-time algorithm for solving the center problem on weighted cactus graphs. *Information Processing Letters*, 71(5-6):205–212, 1999.
- 66 H. M. Mulder and A. Schrijver. Median graphs and Helly hypergraphs. *Discrete Mathematics*, 25(1):41–50, 1979.
- 67 M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13(1):85–108, 1981.
- 68 R. Nowakowski and I. Rival. The smallest graph variety containing all paths. *Discrete Mathematics*, 43(2-3):223–234, 1983.
- 69 S. Olariu. A simple linear-time algorithm for computing the center of an interval graph. *International Journal of Computer Mathematics*, 34(3-4):121–128, 1990.
- 70 L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, pages 515–524, 2013.
- 71 D. Sleator and R. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.
- 72 L. Stewart and R. Valenzano. On polygon numbers of circle graphs and distance hereditary graphs. *Discrete Applied Mathematics*, 248:3–17, 2018.
- 73 S. Ting. A linear-time algorithm for maxisum facility location on tree networks. *Transportation Science*, 18(1):76–84, 1984.
- 74 D. Willard. New data structures for orthogonal range queries. *SIAM Journal on Computing*, 14(1):232–253, 1985.
- 75 P.M. Winkler. Isometric embedding in products of complete graphs. *Discrete Applied Mathematics*, 7(2):221–225, 1984.

On Computing the Average Distance for Some Chordal-Like Graphs

Guillaume Ducoffe  

National Institute of Research and Development in Informatics, Bucharest, Romania
University of Bucharest, Romania

Abstract

The Wiener index of a graph G is the sum of all its distances. Up to renormalization, it is also the average distance in G . The problem of computing this parameter has different applications in chemistry and networks. We here study when it can be done in truly subquadratic time (in the size $n + m$ of the input) on n -vertex m -edge graphs. Our main result is a complete answer to this question, assuming the Strong Exponential-Time Hypothesis (SETH), for all the hereditary subclasses of chordal graphs. Interestingly, the exact same result also holds for the diameter problem. The case of non-hereditary chordal subclasses happens to be more challenging. For the chordal Helly graphs we propose an intricate $\tilde{O}(m^{3/2})$ -time algorithm for computing the Wiener index, where m denotes the number of edges. We complete our results with the first known linear-time algorithm for this problem on the dually chordal graphs. The former algorithm also computes the median set.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Wiener index, Graph diameter, Hardness in P, Chordal graphs, Helly graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.44

Funding *Guillaume Ducoffe*: This work was supported by project PN-19-37-04-01 “New solutions for complex problems in current ICT research fields based on modelling and optimization”, funded by the Romanian Core Program of the Ministry of Research and Innovation (MCI) 2019–2022.

1 Introduction

This paper is about the fine-grained complexity of computing the average distance in a graph, a fundamental distance problem. For any undefined graph terminology, see [4, 20]. Unless stated otherwise, we only consider graphs that are simple, loopless, unweighted undirected, and more importantly connected. Let $G = (V, E)$ be such a graph. Throughout the paper, let $n = |V|$ and $m = |E|$. The distance between two vertices $x, y \in V$ equals the minimum number of edges on a xy -path in G . We denote it by $d_G(x, y)$, or simply by $d(x, y)$ whenever the graph G is clear from the context. The Wiener index of a graph G is $W(G) = \sum_{x, y \in V} d(x, y)$. Let also $\text{diam}(G) = \max_{x, y \in V} d(x, y)$ be the diameter of G . Note that $\text{diam}(G)$ and $\frac{1}{n(n-1)}W(G)$ represent the maximum and average distances in G . Although we focus in this work on computing $W(G)$, as it turns out, this problem is closely related to computing $\text{diam}(G)$. One of our objectives with this paper is to make clearer the connection between both problems.

The study of both parameters has applications in the fields of network optimization and analysis. For instance, delays are amongst the main causes of QoS degradation in a network. Roughly, if we further assume the network is subject to uniformly distributed demand, then we can approximate delays in the networks by distances in the underlying graph. In particular, with this interpretation in mind, the diameter and the (normalized) Wiener index would correspond to the maximum and average delays in the network, respectively. On a different note, for the analysis of social networks, and of more general complex networks with a core-periphery structure, various centrality indices have been introduced in order to measure the importance of a node. One of them, the so-called eccentricity centrality [34],



© Guillaume Ducoffe;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 44; pp. 44:1–44:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

is tightly related to the diameter: in fact, with respect to this centrality measurement, the most peripheral vertices are exactly the diametral vertices (whose eccentricity equals the diameter). In the same way, the Wiener index is related to the closeness centrality [34]. In chemistry, relations were also shown between some quantities of molecules and the Wiener index of their chemical graph representation [42].

By a straightforward reduction to All-Pairs Shortest-Paths (APSP), the Wiener index and the diameter of a graph can both be computed in $\mathcal{O}(nm)$ time. Somehow, this is optimal, even for sparse graphs. Indeed, assuming the Strong Exponential-Time Hypothesis (SETH) [32], one cannot decide in $\tilde{\mathcal{O}}(n^{2-\epsilon})$ time, for any $\epsilon > 0$, whether a graph with $\tilde{\mathcal{O}}(n)$ edges has diameter either 2 or 3 [39]. The former implies that one cannot compute either the Wiener index of such graph in $\tilde{\mathcal{O}}(n^{2-\epsilon})$ time, for any $\epsilon > 0$ [12]. If we allow approximation algorithms, then the situation is completely different. Indeed, while for any fixed ε we can compute an $(1 + \varepsilon)$ -approximation of the Wiener index in almost linear time [27], a long line of recent works has ruled out such possibility for the diameter problem, establishing various trade-off between the allowed running-time and the best possible approximation factor [1, 5, 6, 18, 36]. In what follows, we only consider exact computations, but on restricted graph classes rather than on general graphs.

Let us call an algorithm truly subquadratic if it runs in $\tilde{\mathcal{O}}(n^a m^b)$ time, for some non negative a, b such that $a + b < 2$. On sparse graphs, such running time becomes $\tilde{\mathcal{O}}(n^{a+b})$ which for the Wiener index and the diameter problem is ruled out by SETH. For dense graphs, the running time becomes $\tilde{\mathcal{O}}(n^{a+2b})$, that may be worse than the classic $\mathcal{O}(nm) = \mathcal{O}(n^3)$ -time algorithm for APSP if $a + 2b > 3$. Therefore, so as to avoid this caveat, we are especially interested in running times in $\tilde{\mathcal{O}}(n^a m)$, for some non negative $a < 1$. Algorithms with such running times are known, for the diameter problem, on many graph classes [22, 24]. However, so far, we lack a good picture about the (non)existence of truly subquadratic-time algorithms for the Wiener index and the diameter problems within special graph classes. Indeed, the systematic study of the (non)existence of such algorithms is quite recent, motivated by the hardness results obtained in [7, 39]. Most prior works were about the (non)existence of almost linear-time algorithms, a much more restricted case [17]. We here make progress toward getting such good picture for the subclasses of chordal graphs.

Recall that, in what follows, most of our results apply to the Wiener index. What is remarkable, we think, is that many recent results for the Wiener index were obtained as a byproduct of similar results for the diameter [12, 14, 26]. Said otherwise, many SETH-hardness results for the diameter also apply to the Wiener index and, conversely, many truly subquadratic-time algorithms for computing the diameter can be modified in order to also compute the Wiener index (although this is not the case for all of them, *e.g.*, see [2, 24, 28]). It would be interesting to identify relevant graph classes where the complexity of the Wiener index and the diameter problem are different. One of our results in the paper, obtained for the subclass of chordal Helly graphs, may be a first step in this direction.

Our results. A graph is *chordal* if it has no induced cycle of length at least four. We here propose linear-time and truly subquadratic-time algorithms for computing the Wiener index, on subclasses of chordal graphs and related graph classes. First, in Sec. 2, we consider dually chordal graphs (*a.k.a.*, the clique-graphs of chordal graphs). The former are not chordal graphs in general, but they generalize strongly chordal graphs, and so, directed path graphs and interval graphs [11]. A linear-time algorithm for computing the diameter of dually chordal graphs was presented in [9] (it was recently extended to the computation of all eccentricities, see [21]). We propose a simple linear-time algorithm for computing the Wiener index in this class of graphs (Theorem 1). Doing so, we obtain the first linear-time

algorithm for computing the Wiener index on strongly chordal graphs, and on even larger chordal subclasses such as doubly chordal graphs. Then, in Sec. 3, we design a general divide-and-conquer method on the clique-tree of any chordal graph in order to compute its Wiener index. We give applications of this method in Sec. 4 and 5. Our main result is proved in Sec. 4, where under SETH we completely characterize the hereditary subclasses of chordal graphs for which we can compute the diameter, resp. the Wiener index, in truly subquadratic time. These subclasses turn out to be the same and they can be characterized via a VC-dimension argument or, in more graph-theoretic terms, as those subclasses excluding at least one split graph. See our Theorem 7 for details. Doing so, we get a simple criterion for the existence of truly subquadratic-time algorithms, both for the Wiener index and the diameter, on *many* subclasses of chordal graphs that have been considered in the literature (Corollary 13). Our characterization for the diameter problem follows from several previous works, although to our best knowledge it has not been observed before. Our main technical contribution in Sec. 4 is to prove that the same characterization also holds for the Wiener index. Finally, in Sec. 5, we end up studying the Wiener index for chordal Helly graphs, a prominent non-hereditary subclass of chordal graphs. – Recall that a graph is Helly if any family of pairwise intersecting balls (of arbitrary centers and radii) have a nonempty common intersection. – We state a few open questions in Sec. 6.

Due to lack of space, some proofs are omitted from the following technical sections.

2 Warm-up: Maximum neighbourhood orderings

Let $G = (V, E)$ be a graph. Recall that for a vertex v , $N(v) = \{u \in V \mid uv \in E\}$ denotes its open neighbourhood, while we call $N[v] = N(v) \cup \{v\}$ its closed neighbourhood. A maximum neighbour of a vertex v is some $u \in N[v]$ (possibly, $u = v$) such that $N[w] \subseteq N[u]$ for every $w \in N[v]$. We call a graph G dually chordal if its vertex-set can be totally ordered as (v_1, v_2, \dots, v_n) so that, for every $1 \leq i \leq n$, vertex v_i has a maximum neighbour in the induced subgraph $G \setminus \{v_1, v_2, \dots, v_{i-1}\}$ [11]. Such ordering is sometimes called a MNO (Maximum Neighbourhood Ordering), and it can be computed in linear time [9]. In what follows, we implicitly use the fact that, if a vertex v has a maximum neighbour $u \neq v$, then $G \setminus v$ is an isometric (distance-preserving) subgraph of G .

► **Theorem 1.** *The Wiener index of a dually chordal graph can be computed in linear time.*

Proof. Fix a MNO (v_1, v_2, \dots, v_n) for G . Let $G_0 = G$ and, for every $1 \leq i < n$, let $G_i = G \setminus \{v_1, v_2, \dots, v_i\}$. We observe that, for a vertex in G_{i-1} to be its own maximum neighbour, it must be a universal vertex. Since such a universal vertex can always be chosen last in a MNO of G_{i-1} , we assume from now on that every v_i , $i < n$ has a maximum neighbour $u_i \neq v_i$ in G_{i-1} . In what follows, we scan the ordering once in order to define some variables S_i and functions $\pi_i := V(G_i) \rightarrow \mathbb{N}$. Then, we reverse scan the MNO to compute, for every i and every $u \in V(G_i)$,

$$D_i(u) = S_i + \sum_{w \in V(G_i)} \pi_i(w) \cdot d(u, w).$$

Initially, let $S_0 = 0$ and, for every $v \in V$, let $\pi_0(v) = 1$. Doing so, we ensure that at the end of the algorithm we have $W(G) = \sum_{v \in V} D_0(v)$. Then, let us assume S_{i-1} and π_{i-1} to be known, for some $i > 0$. Let $u_i \in N_{G_{i-1}}(v_i)$ have maximum degree in the (isometric) subgraph G_{i-1} . Note that by maximality of $|N_{G_{i-1}}(u_i)|$, this vertex u_i must be a maximum neighbour of v_i in G_{i-1} . We set $S_i = S_{i-1} + \pi_{i-1}(v_i)$, $\pi_i(u_i) = \pi_{i-1}(u_i) + \pi_{i-1}(v_i)$ and $\pi_i(w) = \pi_{i-1}(w)$ for every other $w \in V(G_i) \setminus \{u_i\}$.

44:4 On Computing the Maximum and Average Distances for Some Chordal-Like Graphs

We now reverse scan the MNO. Clearly, $D_{n-1}(v_n) = S_{n-1}$. Let us assume the values $D_i(u)$ to be known. We set:

$$D_{i-1}(v_i) = D_i(u_i) + n - 2\pi_{i-1}(v_i) - \left(\sum_{w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(w) \right)$$

and we proceed to the following update for every $w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}$: $D_{i-1}(w) = D_i(w) - \pi_{i-1}(v_i)$. Indeed, as it shall become clearer in the remainder of our proof, this update is because these are the only vertices $x \in V(G_i)$ for which we do not have $d_{G_{i-1}}(v_i, x) = d_{G_{i-1}}(u_i, x) + 1$. For every other vertex $x \in V(G_{i-1})$, $D_{i-1}(x) = D_i(x)$.

All the above operations can be performed in total $\sum_i \mathcal{O}(|N_{G_{i-1}}(v_i)|) = \mathcal{O}(m+n)$ time. Furthermore, if all values $D_i(x)$ are correctly computed, then we get $D_0(x) = \sum_{y \in V} d(x, y)$. In particular, $W(G) = \sum_{x \in V} D_0(x)$. Let us assume in what follows all the values $D_i(x)$ to be correctly computed, for some $i > 0$. Since u_i is a maximum neighbour of v_i we have $d(v_i, x) = d(u_i, x) + 1$ for every $x \notin N_{G_{i-1}}[v_i]$. In particular:

$$\begin{aligned} D_{i-1}(x) &= S_{i-1} + \sum_{y \in V(G_{i-1})} \pi_{i-1}(y) \cdot d(y, x) \\ &= S_{i-1} + \pi_{i-1}(v_i) \cdot d(x, v_i) + \pi_{i-1}(u_i) \cdot d(x, u_i) + \sum_{y \in V(G_i) \setminus \{u_i\}} \pi_{i-1}(y) \cdot d(y, x) \\ &= S_{i-1} + \pi_{i-1}(v_i) \cdot (d(x, u_i) + 1) + \pi_{i-1}(u_i) \cdot d(x, u_i) + \sum_{y \in V(G_i) \setminus \{u_i\}} \pi_i(y) \cdot d(y, x) \\ &= S_{i-1} + \pi_{i-1}(v_i) + (\pi_{i-1}(v_i) + \pi_{i-1}(u_i)) \cdot d(x, u_i) + \sum_{y \in V(G_i) \setminus \{u_i\}} \pi_i(y) \cdot d(y, x) \\ &= S_i + \sum_{y \in V(G_i)} \pi_i(y) \cdot d(y, x) = D_i(x). \end{aligned}$$

In the same way (using $d(u_i, u_i) = 0$),

$$\begin{aligned} D_{i-1}(u_i) &= S_{i-1} + \pi_{i-1}(v_i) + \sum_{y \in V(G_{i-1}) \setminus \{v_i\}} \pi_{i-1}(y) \cdot d(y, u_i) \\ &= S_i + \sum_{y \in V(G_i)} \pi_i(y) \cdot d(y, u_i) \\ &= D_i(u_i). \end{aligned}$$

However, for every $w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}$, $\pi_{i-1}(v_i) \cdot d(v_i, w) = \pi_{i-1}(v_i)$ is counted twice in $D_i(w)$: once in S_i , and once in $\pi_i(u_i) \cdot d(u_i, w) = \pi_i(u_i) = \pi_{i-1}(u_i) + \pi_{i-1}(v_i)$. In particular, we obtain $D_{i-1}(w) = D_i(w) - \pi_{i-1}(v_i)$.

We are left proving that $D_{i-1}(v_i)$ is correctly computed. By induction, $\forall j, \sum_x \pi_j(x) = n$. Then, we have:

$$\begin{aligned} D_{i-1}(v_i) &= S_{i-1} + \sum_y \pi_{i-1}(y) \cdot d(y, v_i) = S_i + \left(\sum_y \pi_{i-1}(y) \cdot d(y, v_i) \right) - \pi_{i-1}(v_i) \\ &= S_i + \left(\sum_{w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(w) + \sum_{y \notin N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(y) \cdot (d(y, u_i) + 1) \right) - \pi_{i-1}(v_i) \\ &= S_i + \left(\sum_{y \notin N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(y) \cdot d(y, u_i) \right) + \left(\sum_{y \neq v_i} \pi_{i-1}(y) \right) - \pi_{i-1}(v_i) \end{aligned}$$

$$\begin{aligned}
&= S_i + \left(\sum_{y \notin N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_i(y) \cdot d(y, u_i) \right) + n - 2\pi_{i-1}(v_i) \\
&= S_i + \left(\sum_y \pi_i(y) \cdot d(y, u_i) \right) - \left(\sum_{w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(w) \right) + n - 2\pi_{i-1}(v_i) \\
&= D_i(u_i) - \left(\sum_{w \in N_{G_{i-1}}(v_i) \setminus \{u_i\}} \pi_{i-1}(w) \right) + n - 2\pi_{i-1}(v_i). \quad \blacktriangleleft
\end{aligned}$$

A vertex v is a median if it minimizes $\sum_{u \in V} d(u, v)$. The median set of a graph G contains all its medians. With the same proof as for Theorem 1, we obtain:

► **Corollary 2.** *The median set of a dually chordal graph can be computed in linear time.*

Finally, a graph G is doubly chordal if it is both chordal and dually chordal [37]. Note that doubly chordal graphs properly contain the strongly chordal graphs, and so, the directed path graphs and the interval graphs.

► **Corollary 3.** *The Wiener index and the median set of a doubly chordal graph (and so, of a strongly chordal graph, resp. directed path graph, resp. interval graph) can be computed in linear time.*

We refer to [19] for a previous linear-time algorithm for the interval graphs. In contrast to our own algorithm, the former is taking an interval representation of the graph as input.

3 A framework for chordal graphs

We introduce a general method for computing the Wiener index of chordal graphs. We recall that a graph is called a split graph if its vertex-set can be bi-partitioned into a clique and a stable set [29]. In the SPLIT-WEIGHTED-WIENER problem, we are given as input a tuple $(\mathcal{P}(V), E', \alpha, \beta)$ where, for some split graph $H = (V, E)$:

- $\mathcal{P}(V) = (K, S^1, S^2, \dots, S^c)$ is a partition of the vertex-set V , where K is a clique and $S := \bigcup_{i=1}^c S^i$ is a stable set.
- $E' = \{uv \mid u \in K, v \in S\} \subseteq E$.
- $\alpha, \beta : V \rightarrow \mathbb{N}_{\geq 1}$ are weight functions.

We call H the underlying input split graph.

The output is equal to $\sum_{i \neq j} \sum_{x \in S^i, y \in S^j} [\beta(y)\alpha(x) + \beta(x)\beta(y)d(x, y) + \beta(x)\alpha(y)]$.

► **Theorem 4.** *There is an $\tilde{O}(m + n)$ -time reduction from computing the Wiener index on a chordal graph G to the SPLIT-WEIGHTED-WIENER problem on some instances $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$. Furthermore, each underlying split H_k is obtained from some induced subgraph of G by removing the edges with their both ends in the same group S_k^i of the partition.*

We need to introduce a few additional notions and related intermediate results.

First, recall that a clique-tree of a graph G is a tree T of which the nodes are the maximal cliques of G , and such that for every vertex v the set of all the maximal cliques that contain v induces a connected subtree. It is known that G is chordal if and only if it has a clique-tree [13, 30, 41] and, furthermore, a clique-tree can be computed in linear time [40]. – See also [3, 10] and the references therein. – We may see a clique-tree T as a node-weighted tree where, for any maximal clique C , $w(C) = |C|$. Then, let $w(T) := \sum_C w(C)$. For a chordal graph, $w(T) = \mathcal{O}(n + m)$ [3].

For a set S and a vertex x , let us define $Pr(x, S) = \{y \in S \mid d(x, y) = d(x, S)\}$. Let also $I(x, y) = \{z \in V \mid d(x, y) = d(x, z) + d(z, y)\}$ for every vertices x and y . The following two results will be useful in our proofs:

► **Lemma 5** ([15]). *In a chordal graph G , if C is a clique and $x \notin C$, then there exists a vertex $g(x) \in \bigcap \{I(x, y) \mid y \in Pr(x, C)\}$ that is adjacent to all vertices from $Pr(x, C)$. This vertex $g(x)$ is sometimes called a gate of x .*

► **Lemma 6** ([22]). *If T is a clique-tree of a chordal graph G then, for every (not necessarily maximal) clique C of G , for every $v \notin C$ we can compute $d_G(v, C)$ and a corresponding gate v^* in total $\mathcal{O}(w(T))$ time, where $w(T)$ denotes the sum of cardinalities of all the maximal cliques of G .*

Finally, for an n -node tree $T = (V, E)$, a centroid is a node whose removal leaves subtrees of order at most $n/2$. A classic theorem from Jordan asserts that such node always exists [33]. Furthermore, we can compute a centroid in $\mathcal{O}(n)$ time by dynamic programming (e.g., see [31]).

Proof of Theorem 4. Up to additional $\mathcal{O}(n + m)$ -time pre-processing, we may assume each input graph G to be given under the form of a clique tree T . Our reduction is recursive. Consider first the following two base cases:

- **Case $|V(T)| = 1$.** Then, G is a clique, and we have $W(G) = n(n - 1)$.
- **Case $|V(T)| = 2$.** Then, G is the union of two intersecting cliques X and X' . In particular, $diam(G) = 2$, and so (see [12]), we have $W(G) = 2n(n - 1) - 2m$. Note that $n = |X| + |X'| - |X \cap X'|$ and $2m = |X| \cdot (|X| - 1) + |X'| \cdot (|X'| - 1) - |X \cap X'| \cdot (|X \cap X'| - 1)$ (computable by scanning once the maximal cliques of G).

From now on, $|V(T)| \geq 3$. We compute X a centroid of T . Let T_1, T_2, \dots, T_c be the subtrees of $T \setminus \{X\}$. For each i , let G_i be induced by the vertices contained in at least one node of T_i . By the properties of a clique-tree, each G_i is an isometric subgraph of G .

(1) Computation of the $W(G_i)$'s. We apply our reduction to G_1, G_2, \dots, G_c (encoded by their respective clique-trees T_1, T_2, \dots, T_c). Doing so (throughout one-to-many reductions to the SPLIT-WEIGHTED-WIENER problem), we computed their respective Wiener indices $W(G_1), W(G_2), \dots, W(G_c)$.

(2) Computation of a first (non definitive) instance $(\mathcal{P}, E', \alpha, \beta)$. We apply Lemma 6 in order to compute, for every $v \notin X$, $d(v, X)$ and a gate $g(v)$, whose existence is ensured by Lemma 5. For every i , let $U_i = \{g(v_i) \mid v_i \in V(G_i) \setminus X\}$. We set, for every $u_i \in U_i$:

$$\alpha(u_i) = \sum \{d(v_i, X) - 1 \mid g(v_i) = u_i\}, \text{ and } \beta(u_i) = \#\{v_i \in V(G_i) \setminus X \mid g(v_i) = u_i\}.$$

We consider the following instance $(\mathcal{P}, E', \alpha, \beta)$ where we have:

- $\mathcal{P} = (X, U_1, U_2, \dots, U_c)$;
- $E' = E \cap (X \times V \setminus X)$ (edges between X and the neighbours of X);
- and α, β as they were previously defined in the proof.

We claim that by solving the SPLIT-WEIGHTED-WIENER problem on this above instance, one computes the sum of all distances $d(v_i, v_j)$ for $v_i \in V(G_i) \setminus X$, $v_j \in V(G_j) \setminus X$ and $i \neq j$. Indeed, let $v_i \in V(G_i) \setminus X$ and $v_j \in V(G_j) \setminus X$, for some $i \neq j$. Every $v_i v_j$ -path crosses X . Furthermore, since X is a clique, there is a shortest $v_i v_j$ -path such that the vertex of X closest to v_i (resp., to v_j) is in $Pr(v_i, X)$ (resp., in $Pr(v_j, X)$). As a result, we have $d(v_i, v_j) = d(v_i, g(v_i)) + d(g(v_i), g(v_j)) + d(g(v_j), v_j)$. For v_i fixed, we get:

$$\begin{aligned}
\sum_{v_j \in V(G_j) \setminus X} d(v_i, v_j) &= \sum_{v_j \in V(G_j) \setminus X} (d(v_i, g(v_i)) + d(g(v_i), g(v_j)) + d(g(v_j), v_j)) \\
&= \sum_{u_j \in U_j} \sum_{v_j | g(v_j) = u_j} (d(v_i, g(v_i)) + d(g(v_i), u_j) + d(u_j, v_j)) \\
&= \sum_{u_j \in U_j} (\beta(u_j) \cdot (d(v_i, g(v_i)) + d(g(v_i), u_j)) + \alpha(u_j)).
\end{aligned}$$

If we sum the above over all the v_i 's, we obtain:

$$\begin{aligned}
\sum_{v_i \in V(G_i) \setminus X} \sum_{v_j \in V(G_j) \setminus X} d(v_i, v_j) &= \sum_{v_i \in V(G_i) \setminus X} \sum_{u_j \in U_j} (\beta(u_j) \cdot (d(v_i, g(v_i)) + d(g(v_i), u_j)) + \alpha(u_j)) \\
&= \sum_{u_i \in U_i} \sum_{u_j \in U_j} (\beta(u_j) \cdot (\alpha(u_i) + \beta(u_i)d(u_i, u_j)) + \beta(u_i)\alpha(u_j)) \\
&= \sum_{u_i \in U_i} \sum_{u_j \in U_j} (\beta(u_j)\alpha(u_i) + \beta(u_j)\beta(u_i)d(u_i, u_j) + \beta(u_i)\alpha(u_j)).
\end{aligned}$$

(3) Computation of a reduced instance. The problem with the above instance $(\mathcal{P}, E', \alpha, \beta)$ is that, in order to fit with our claimed running time for the reduction, we further need to have $|E'| = \mathcal{O}(w(T))$, that may not be the case in general. Thus, we need to reduce the instance. For that, let $U := \bigcup_i U_i$. For every $u \in U$, there is a maximal clique that contains $\{u\} \cup (N(u) \cap X)$. Thus, in order to relate u with its neighbours in X , it suffices to compute amongst all maximal cliques containing u one X_u maximizing $|X_u \cap X|$. We can do so, in $\mathcal{O}(w(T))$ time, as follows:

- We scan all the maximal cliques $X' \neq X$ in order to compute $|X' \cap X|$.
- We order the maximal cliques $X' \neq X$ by non increasing value of $|X' \cap X|$. It can be done by using, *e.g.*, counting sort.
- We scan all the ordered maximal cliques X' . Initially, all the vertices are left unmarked. When scanning a maximal clique X' , we set $X_u = X'$ for every $u \in X' \cap U$ unmarked. Then, we mark all vertices in X' .

Let $u, u' \in U$ be such that $X_u = X_{u'}$. Since u and u' are adjacent, there is an i such that $u, u' \in U_i$. Since furthermore, $N(u) \cap X = N(u') \cap X$ (these vertices are twins in the underlying split graph H), we may remove u' from U and update $\alpha(u), \beta(u)$ as follows: $\alpha(u) := \alpha(u) + \alpha(u')$, $\beta(u) := \beta(u) + \beta(u')$. Doing so until it can no more be done, we end up with a smaller set U^* such that no two vertices $u \in U^*$ are associated to the same maximal clique X_u . For every i , let us replace U_i in the above instance for SPLIT-WEIGHTED-WIENER by the subset $S_i = U_i \cap U^*$. Then, for $E^* = \{uv \mid u \in U^*, v \in X\}$ we obtain $|E^*| < \sum_{u \in U^*} |X_u| = \mathcal{O}(w(T))$, as desired.

From now on, we assume to be given $W_0 := \sum_{i \neq j} \sum_{v_i \in V(G_i) \setminus X, v_j \in V(G_j) \setminus X} d(v_i, v_j)$ (*i.e.*, by the above reduction to one instance of SPLIT-WEIGHTED-WIENER).

(4) Computation of the sum of distances between X and $V \setminus X$. Next, we compute $W_X := \sum_{x \in X, v \notin X} d(x, v)$. For that, recall that we computed the set $U = \bigcup_i U_i$ of all the gates, and the weight functions α, β . Furthermore, for each $x \in X$ and index i we have:

$$\sum_{v_i \in V(G_i) \setminus X} d(x, v_i) = \sum_{v_i \in V(G_i) \setminus X} (d(x, g(v_i)) + d(g(v_i), v_i)) = \sum_{u_i \in U_i} (\beta(u_i)d(x, u_i) + \alpha(u_i)).$$

Note that, for every $u \in U$, $d(u, x) \in \{1, 2\}$. Hence, we get:

$$\sum_{v \notin X} d(x, v) = 2 \cdot \left(\sum_{u \in U} \beta(u) \right) - \left(\sum_{u \in N(x) \cap U} \beta(u) \right) + \left(\sum_{u \in U} \alpha(u) \right).$$

The two sums $\sum_{u \in U} \beta(u)$ and $\sum_{u \in U} \alpha(u)$ can be pre-computed in $\mathcal{O}(|U|) = \mathcal{O}(w(T))$ time. Therefore, in order to compute W_X , we are left computing $\sum_{u \in N(x) \cap U} \beta(u)$ for every $x \in X$.

- For each $x \in X$, let $\gamma(x) = 0$ (at the end of the procedure, we shall have $\gamma(x) = \sum_{u \in N(x) \cap U} \beta(u)$).
- We root T arbitrarily and we start a BFS from the root.
- When we reach some maximal clique X' during the search, we further assume to have access to its intersection $Y = X' \cap p(X')$ with its parent node ($Y = \emptyset$ if X' is the root). For every $x \in (X \cap X') \setminus Y$, we increment $\gamma(x)$ by $\sum_{u \in U \cap X'} \beta(u)$. However, in order to avoid overcounting, for every $x \in X \cap Y$, we increment $\gamma(x)$ by $\sum_{u \in (U \cap X') \setminus Y} \beta(u)$.
- After processing each X' , we scan all the maximal cliques X'' that are children nodes of X' in order to compute $X' \cap X''$.

Since each maximal clique is scanned $\mathcal{O}(1)$ times, the total running time is in $\mathcal{O}(w(T))$.

(5) A formula for computing $W(G)$. At this point of the reduction, we are almost done for computing $W(G)$. However, if we sum all the partial estimates computed so far, there are a few distances overcounted. Specifically, let $Y_i = X \cap V(G_i)$. We have:

$$\begin{aligned} W(G) &= W_0 + \left(\sum_{i=1}^c W(G_i) \right) + |X| \cdot (|X| - 1) + 2W_X \\ &\quad - \sum_{i=1}^c \left(|Y_i| \cdot (|Y_i| - 1) + 2 \cdot \sum_{y_i \in Y_i} \sum_{v_i \in V(G_i) \setminus X} d(v_i, y_i) \right) \end{aligned}$$

Each set Y_i above can be computed as the intersection between X and the unique maximal clique $X_i \in V(T_i) \cap N_T(X)$. Furthermore, for $y_i \in Y_i$ fixed, we have:

$$\sum_{v_i \in V(G_i) \setminus X} d(v_i, y_i) = 2 \cdot \left(\sum_{u_i \in U_i} \beta(u_i) \right) - \left(\sum_{u_i \in N(y_i) \cap U_i} \beta(u_i) \right) + \left(\sum_{u_i \in U_i} \alpha(u_i) \right).$$

Hence, we are left computing $\gamma_i(y_i) = \sum_{u_i \in U_i \cap N(y_i)} \beta(u_i)$ for every $y_i \in Y_i$. This can be done in $\mathcal{O}(w(T_i))$ time, by using the same procedure as for computing the values $\gamma(x)$, but restricted to the clique-subtree T_i . Since all the T_i 's are disjoint, the total running time is still in $\mathcal{O}(w(T))$.

Complexity. Since we use in our reduction a centroid decomposition of the clique-tree T , there are $\mathcal{O}(\log |V(T)|) = \mathcal{O}(\log n)$ recursive stages. At each recursive stage, we proceed on disjoint clique-subtrees T' . Therefore, each recursive stage takes $\mathcal{O}(w(T))$ time (excluding the calls to an oracle solving SPLIT-WEIGHTED-WIENER). The total running time of the reduction, excluding the calls of the oracle, is in $\mathcal{O}(w(T) \log n) = \mathcal{O}(m \log n)$. ◀

We shall use the heavy machinery presented above in the next two sections.

4 Application: Hereditary subclasses of chordal graphs

Recall that a class of graphs is called hereditary if it is stable by induced subgraph. The complexity of the diameter problem has been studied for many hereditary subclasses of chordal graphs [9, 17, 25, 22, 23, 38]. For the special case of the interval graphs, a linear-time algorithm for computing the Wiener index is also known [19] (which we extended to the strongly chordal graphs in Sec. 2). However, even for the hereditary subclass of split graphs, under SETH there is no truly subquadratic algorithm for computing the diameter nor the Wiener index. In this section, we exactly characterize the hereditary subclasses of chordal graphs for which such algorithms exist (conditioned on SETH).

Recall that a hypergraph is a pair $\mathcal{H} = (X, \mathcal{R})$ such that each element of \mathcal{R} (called a hyperedge) is a subset of X (the elements of X are called vertices, by analogy to graphs). A vertex-subset $Y \subseteq X$ is shattered by \mathcal{H} if, for any possible subset $Z \subseteq Y$, there exists an $e \in \mathcal{R}$ such that $e \cap Y = Z$. The VC-dimension of \mathcal{H} is the largest cardinality of a shattered subset. For a graph, its VC-dimension is the VC-dimension of its neighbourhood hypergraph $\mathcal{N}(G) = (V, \{N[v] \mid v \in V\})$. Finally, a class of graphs has bounded VC-dimension if there exists a constant d such that every graph in the class has VC-dimension at most d .

► **Theorem 7.** *Under SETH, for any hereditary subclass \mathcal{C} of chordal graphs, the following statements are equivalent:*

1. *There is a truly subquadratic algorithm for computing the Wiener index within \mathcal{C} .*
2. *There is a truly subquadratic algorithm for computing the diameter within \mathcal{C} .*
3. *There is a truly subquadratic algorithm for deciding if a graph in \mathcal{C} has diameter ≤ 2 .*
4. *\mathcal{C} does not contain all the split graphs.*
5. *\mathcal{C} has bounded VC-dimension.*

The above theorem follows from previous works in the literature, and a new result on our own in Sec. 4.1. Specifically, we will use the following lemmas in our proof:

► **Lemma 8** ([7]). *For any $\varepsilon > 0$, there exists a $c(\varepsilon)$ s.t., under SETH, we cannot compute the diameter in $\mathcal{O}(n^{2-\varepsilon})$ time on the split graphs of order n and clique-number at most $c(\varepsilon) \log n$.*

► **Lemma 9** ([22]). *If \mathcal{C} is a subclass of chordal graphs of bounded VC-dimension, then there exists a randomized truly subquadratic-time algorithm for computing the diameter of the graphs in \mathcal{C} .*

The next result of Bousquet et al. [8] shows that for any hereditary chordal subclass, either Lemma 8 or Lemma 9 can be applied.

► **Lemma 10** ([8]). *Let \mathcal{C} be a hereditary class. If \mathcal{C} has infinite VC-dimension, then \mathcal{C} must contain either all the bipartite graphs, or all the co-bipartite graphs, or all the split graphs.*

We also prove a quantitative version of Lemma 10, for chordal graphs. Note that the worst-case running time of the algorithms presented in Lemma 9 and in Sec. 4.1 depends on the largest VC-dimension of a graph in the class \mathcal{C} .

► **Lemma 11.** *If H is a split graph, then every H -free chordal graph has VC-dimension at most $|V(H)| - 1$.*

In Sec. 4.1, we prove that:

► **Theorem 12.** *If \mathcal{C} is a subclass of chordal graphs of bounded VC-dimension, then there exists a deterministic truly subquadratic-time algorithm for computing the Wiener index of the graphs in \mathcal{C} .*

We are finally ready to prove our main result:

Proof of Theorem 7. Let $G \in \mathcal{C}$ be arbitrary. It is known [12] that $W(G) \leq 2n(n-1) - 2m$ if and only if $\text{diam}(G) \leq 2$. Therefore, (1) \implies (3). We also have (2) \implies (3). Since the diameter of a split graph is at most three, we get by Lemma 8 that (3) \implies (4). Furthermore, since not all bipartite graphs and co-bipartite graphs are chordal, by Lemma 10 we get that (4) \implies (5). Finally, by Lemma 9 we have (5) \implies (2), and by Theorem 12 we have (5) \implies (1). ◀

► **Corollary 13.** *The following subclasses of chordal graphs admit truly subquadratic algorithms for the Wiener index and the diameter problem: chordal bull-free graphs, chordal claw-free graphs, block graphs, interval graphs [38], strongly chordal graphs [9], directed path graphs [17], undirected path graphs [22], chordal dominating pair graphs [25], hereditary Helly graphs [22], k -separator chordal graphs [35], chordal graphs of bounded interval number, chordal graphs of bounded asteroidal number [25].*

To our best knowledge, our results are new for chordal bull-free graphs, chordal claw-free graphs, k -separator chordal graphs and chordal graphs of bounded interval number, both for the Wiener index and the diameter problem (references to prior works are given in the statement of Corollary 13). For the Wiener index only, our results are also new for the subclasses of strongly chordal graphs (see also Sec. 2 for a faster algorithm), directed path graphs, undirected path graphs, chordal dominating pair graphs, hereditary Helly graphs and chordal graphs of bounded asteroidal number. This above listing is far from exhaustive.

4.1 Sketch Proof of Theorem 12

In what follows, let $G = (V, E) \in \mathcal{C}$. By Theorem 4, computing the Wiener index of G can be reduced in $\tilde{O}(m+n)$ time to solving the SPLIT-WEIGHTED-WIENER problem on some instances $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$. Let d be the maximum VC-dimension of a graph in \mathcal{C} . We prove below that each instance $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$ can be solved in $\tilde{O}(|E'_k| \cdot |V_k|^{1-\varepsilon_d})$ time, where ε_d is a constant that only depends on d . Note that $\sum_k |E'_k| = \tilde{O}(m+n)$ since it is the total running time of our reduction. Furthermore, since by Theorem 4 each V_k is a subset of V , $\max_k |V_k| = \mathcal{O}(n)$. Hence, our result below implies an $\tilde{O}(\sum_k |E'_k| \cdot |V_k|^{1-\varepsilon_d}) = \tilde{O}(mn^{1-\varepsilon_d})$ running time in order to compute $W(G)$.

For the remainder of the proof, let $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$ be fixed. Recall (see Sec. 3) $\mathcal{P}(V_k) = (K_k, S_k^1, S_k^2, \dots, S_k^{c_k})$ with K_k a clique of G . Let $S_k = \bigcup_j S_k^j = V_k \setminus K_k$. By Theorem 4, $E'_k = E(G) \cap (K_k \times S_k)$ (*i.e.*, there is no edge added or removed between K_k and S_k compared to G). We start with a simple observation:

► **Lemma 14.** *Let $G = (V, E)$ have VC-dimension at most d , and let $X, Y \subseteq V$. The hypergraph $\mathcal{H} = (X, \{N_G[y] \cap X \mid y \in Y\})$ also has VC-dimension at most d .*

Proof. Any subset shattered by \mathcal{H} is shattered by the neighbourhood hypergraph of G . ◀

We apply Lemma 14 to $X = S_k$ and $Y = K_k$. Let \mathcal{H}_k be the corresponding hypergraph.

A spanning path of a hypergraph \mathcal{H} is a total order of its vertex-set. The stabbing number of such spanning path is the least t such that every hyperedge of \mathcal{H} is the union of at most t intervals onto the path. The following result is based on a prior work of Chazelle and Welzl about range queries [16]:

► **Lemma 15** ([24]). *For every $d > 0$, there exists a constant $\varepsilon_d \in (0; 1)$ such that in $\tilde{\mathcal{O}}(m+n^{2-\varepsilon_d})$ deterministic time, for every n -vertex hypergraph $\mathcal{H} = (X, \mathcal{R})$ of VC-dimension at most d and size $m = \sum_{e \in \mathcal{R}} |e|$, we can compute a spanning path of stabbing number $\tilde{\mathcal{O}}(n^{1-\varepsilon_d})$. Moreover, $\varepsilon_d = \frac{1}{2^{d+1} \lceil c(d+1) - 1 \rceil + 1}$ for some constant $c > 2$.*

Apply Lemma 15 to \mathcal{H}_k . Doing so, for every $u \in K_k$, $N_H(u) \cap S_k$ is the union of $\tilde{\mathcal{O}}(|S_k|^{1-\varepsilon_d})$ intervals of the resulting spanning path of \mathcal{H}_k . Then, for every $s \in S_k$, define $N_k^2(s)$ to be the set of all vertices in S_k at distance two from s in the underlying split graph $(K_k \cup S_k, E'_k \cup (K_k \times K_k))$. We have that $N_k^2(s)$ is the union of $\tilde{\mathcal{O}}(|N_G(s) \cap K_k| \cdot |S_k|^{1-\varepsilon_d})$ intervals of the spanning path computed for \mathcal{H}_k .

One more ingredient is needed in our proof. Consider a set Q of 2-dimensional points. Each point $(x, y) \in Q$ is assigned some weight $f(x, y)$. A box is the Cartesian product of two intervals (we also allow intervals that are infinite, semi-finite, or reduced to a singleton). Note that each box defines a rectangle in the plane (possibly, a line or a point if some intervals are reduced to a singleton). A (counting) range query asks, for a given box, the sum of the weights of all points in Q that are contained into this rectangle.

► **Lemma 16** ([43]). *Let Q be a set of 2-dimensional points. After a pre-processing in $\mathcal{O}(|Q| \log |Q|)$ time, one can answer any range query in $\mathcal{O}(\log |Q|)$ time.*

Let $\sigma_k : S_k \rightarrow \{1, 2, \dots, |S_k|\}$ be the mapping of S_k to the nodes of the spanning path. For each $1 \leq i \leq c_k$ and $s_i \in S_k^i$, we create a point $(\sigma_k(s_i), i)$ with weight $f(\sigma_k(s_i), i) = \beta_k(s_i)$. Let Q_k be the resulting 2-dimensional point-set. We apply Lemma 16 in order to compute, for each i and $s_i \in S_k^i$, the weighted sum $\Phi(s_i) = \sum \{\beta_k(s') \mid s' \in N_k^2(s_i) \setminus S_k^i\}$; indeed, for any fixed s_i , this operation can be reduced to $\tilde{\mathcal{O}}(|N_G(s_i) \cap K_k| \cdot |S_k|^{1-\varepsilon_d})$ range queries, by using the interval representation of $N_k^2(s_i)$.

Finally, for every j , define $\alpha^j = \sum_{s' \in S_k^j} \alpha_k(s')$ and $\beta^j = \sum_{s' \in S_k^j} \beta_k(s')$. Let also $\alpha^* = \sum_{j=1}^{c_k} \alpha^j$ and $\beta^* = \sum_{j=1}^{c_k} \beta^j$. All the values α^j, β^j , $1 \leq j \leq c_k$ and α^*, β^* can be pre-computed in total $\mathcal{O}(|S_k|)$ time. Since in the underlying split graph H_k , the distance $d(s, s')$ between two different vertices $s, s' \in S_k$ in the stable set is either two or three, we may rewrite the output of SPLIT-WEIGHTED-WIENER as follows:

$$\begin{aligned} & \sum_{i \neq j} \sum_{s \in S_k^i, s' \in S_k^j} [\beta_k(s') \alpha_k(s) + \beta_k(s) \beta_k(s') d(s, s') + \beta_k(s) \alpha_k(s')] = \\ & \sum_i \sum_{s \in S_k^i} \left[(\beta^* - \beta^i) \cdot \alpha_k(s) + \beta_k(s) \cdot \left(\sum_{s' \notin S_k^i} \beta_k(s') d(s, s') \right) + \beta_k(s) \cdot (\alpha^* - \alpha^i) \right] \\ & = \left[\sum_i (\beta^* - \beta^i) \alpha^i \right] + \left[\sum_i \sum_{s \in S_k^i} \beta_k(s) \cdot \left(\sum_{s' \notin S_k^i} \beta_k(s') d(s, s') \right) \right] + \left[\sum_i \beta^i (\alpha^* - \alpha^i) \right] \\ & = \left[\sum_i (\beta^* - \beta^i) \alpha^i \right] + \left[\sum_i \sum_{s \in S_k^i} \beta_k(s) \cdot (3(\beta^* - \beta^i) - \Phi(s)) \right] + \left[\sum_i \beta^i (\alpha^* - \alpha^i) \right] \\ & = \left[\sum_i (\beta^* - \beta^i) (\alpha^i + 3\beta^i) \right] - \left[\sum_{s \in S_k} \beta_k(s) \Phi(s) \right] + \left[\sum_i \beta^i (\alpha^* - \alpha^i) \right]. \end{aligned}$$

Now, given the pre-computed values $\alpha^i, \beta^i, \alpha^*, \beta^*$ and $\Phi(s)$, we can compute the desired output in additional $\mathcal{O}(|S_k|)$ time. ◀

5 Application: Chordal Helly graphs

We end up our study investigating the complexity of the Wiener index on non-hereditary subclasses of chordal graphs. The chordal Helly graphs are a prominent such subclass. Indeed, they are a strict generalization of doubly chordal graphs, and so, of strongly chordal graphs, interval graphs, etc. Recently, a linear-time algorithm for computing the diameter of this subclass of graphs was proposed [22].

► **Theorem 17.** *There is an $\tilde{O}(m^{3/2})$ -time algorithm for computing the Wiener index of chordal Helly graphs.*

Proof. Let $G = (V, E)$ be a chordal Helly graph. We apply Theorem 4 in order to reduce the computation of $W(G)$ to solving the SPLIT-WEIGHTED-WIENER problem on some instances $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$. Write $\mathcal{P}(V_k) = (K_k, S_k^1, S_k^2, \dots, S_k^{c_k})$. We also know from Theorem 4 that K_k is a clique of G . In what follows, we present an algorithm for solving the instance $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$ in $\mathcal{O}(|K_k| \cdot |E'_k|)$ time. Doing so, we can compute $W(G)$ in $\sum_k \mathcal{O}(|K_k| \cdot |E'_k|)$ time. Since we further have $\sum_k |E'_k| = \tilde{O}(n + m)$ (time of the reduction of Theorem 4) and $\max_k |K_k| = \mathcal{O}(m^{1/2})$ because each subset K_k is a clique, we get a running time in $\tilde{O}(m^{3/2})$.

Throughout the remainder of the proof, let $(\mathcal{P}(V_k), E'_k, \alpha_k, \beta_k)$ be fixed. Let also $S_k = \bigcup_{i=1}^{c_k} S_k^i = V_k \setminus K_k$. We write $s \sim s'$ if there exists an i such that $s, s' \in S_k^i$. Then, our goal is to compute the following value:

$$\Psi_k := \sum \{\beta_k(s)\beta_k(s') \mid s \not\sim s' \text{ and } d(s, s') = 2\}.$$

Indeed, let us define $\alpha^i = \sum_{s \in S_k^i} \alpha_k(s)$ and $\beta^i = \sum_{s \in S_k^i} \beta_k(s)$ for every i . Similarly, let $\alpha^* = \sum_i \alpha^i$ and $\beta^* = \sum_i \beta^i$. All these values can be pre-computed in total $\mathcal{O}(|S_k|) = \mathcal{O}(|E'_k|)$ time. Then, the desired output $\sum_{i \neq j} \sum_{s \in S_k^i, s' \in S_k^j} [\beta_k(s')\alpha_k(s) + \beta_k(s)\beta_k(s')d(s, s') + \beta_k(s)\alpha_k(s')]$ can be rewritten as:

$$\sum_i (\beta^* - \beta^i)\alpha^i + \left(\sum_{i \neq j} \sum_{s \in S_k^i, s' \in S_k^j} \beta_k(s)\beta_k(s')d(s, s') \right) + \sum_i \beta^i(\alpha^* - \alpha^i).$$

Since in the underlying split graph, the distance between two distinct vertices in the stable set S_k is either two or three, we get:

$$\sum_{i \neq j} \sum_{s \in S_k^i, s' \in S_k^j} \beta_k(s)\beta_k(s')d(s, s') = 3 \left(\sum_i \beta^i(\beta^* - \beta^i) \right) - \Psi_k.$$

The algorithm. Let us define the adjacency lists $N_k(u) = \{s \in S_k \mid us \in E'_k\}$, for every $u \in K_k$. In the same way, let us define the adjacency lists $N_k(s) = \{u \in K_k \mid us \in E'_k\}$, for every $s \in S_k$. We set initially $\Psi_k := 0$. Then, let $K_k = (u_1, u_2, \dots, u_{|K_k|})$ be totally ordered. We consider each $u_i \in K_k$ sequentially, and in order from $i = 1$ to $i = |K_k|$. At step i , let us define for every $u' \in K_k$ the subset $N_{k,i}(u') := N_k(u') \setminus \left(\bigcup_{j < i} N_k(u_j) \right)$. For each $s \in S_k \setminus \left(\bigcup_{j < i} N_k(u_j) \right)$, let t be such that $s \in S_k^t$. We select a vertex $u' \in N_k(s)$ such that $|(N_{k,i}(u') \cap N_{k,i}(u_i)) \setminus S_k^t|$ is maximized. Then, we increment Ψ_k by:

$$\begin{cases} \beta_k(s) \cdot \sum \{\beta_k(s') \mid s' \in (N_{k,i}(u_i) \cap N_{k,i}(u')) \setminus S_k^t\} & \text{if } s \in N_{k,i}(u_i) \\ 2\beta_k(s) \cdot \sum \{\beta_k(s') \mid s' \in (N_{k,i}(u_i) \cap N_{k,i}(u')) \setminus S_k^t\} & \text{if } s \notin N_{k,i}(u_i). \end{cases}$$

Correctness. We first need to observe that $N_{k,1}(u_1)$ ($= N_k(u_1)$), $N_{k,2}(u_2), \dots, N_{k,i}(u_i), \dots$ is a partition of S_k . Therefore in order to prove correctness of the algorithm, it suffices to prove that at each step i , we increment Ψ_k by twice the sum of all $\beta_k(x)\beta_k(y)$ with $x \in N_{k,i}(u_i)$, $y \in S_k \setminus \left(\bigcup_{j < i} N_k(u_j)\right)$, $x \not\sim y$ and $d(x, y) = 2$. For that, let $s \in S_k \setminus \left(\bigcup_{j < i} N_k(u_j)\right)$ be arbitrary and such that $s \in S_k^t$ for some t . We prove next that for the vertex u' selected for s , all the vertices in $N_{k,i}(u_i) \setminus S_k^t$ and at distance two from s are also in $N_{k,i}(u')$. In particular, all the desired pairs (x, y) are enumerated: twice if $x, y \in N_{k,i}(u_i)$, and only once otherwise, thus proving correctness of our above formula for incrementing Ψ_k .

By maximality of u' , it is sufficient to prove the existence of a vertex $u^* \in N_k(s)$ such that all the vertices of $N_{k,i}(u_i) \setminus S_k^t$ that are at distance two from s are also contained into $N_{k,i}(u^*)$. We do so by reasoning on the whole graph G . Specifically, let $\mathcal{F} = \{N_G[x] \mid x = s \text{ or } (x \in N_{k,i}(u_i) \setminus S_k^t \text{ and } d(s, x) = 2)\}$. The balls in \mathcal{F} pairwise intersect. Therefore, by the Helly property, all balls in \mathcal{F} contain some common vertex z . We claim that $z \in K_k$. Note that it will prove the existence of the desired vertex u^* because in such a case we can always choose $u^* = z$. It follows from the proof of Theorem 4 that S_k^t and $S_k \setminus S_k^t$ are in separate connected components of $G \setminus K_k$, thus immediately proving the claim.

Implementation and Complexity. As a starter, we observe that all the lists $N_k(u)$, for $u \in K_k$ (resp., all the lists $N_k(s)$, for $s \in S_k$) can be constructed in $\mathcal{O}(|E'_k|)$ time. Throughout the algorithm, we maintain an $|S_k|$ -size array \mathbf{A} , whose entries are indexed by S_k and are initialized to 0 (at the end of the algorithm, we have for all $s \in N_{k,i}(u_i)$ that $\mathbf{A}[s] = i$). We also store two auxiliary matrices of dimensions $|K_k| \times (c_k + 1)$, denoted by \mathbf{Int} and \mathbf{Sum} , whose entries are indexed by $K_k \times \{0, 1, 2, \dots, c_k\}$ and of which we ensure that all entries equal 0 at the beginning of any step i . Finally, we find more convenient for certain operations to maintain a stack \mathbf{Stack} and a boolean $|K_k|$ -size array $\mathbf{InStack}$ whose entries are indexed by K_k ; before each step, we ensure that the stack is emptied and that all entries in $\mathbf{InStack}$ are set to False. Note that all the above data structures can be constructed in $\mathcal{O}(|S_k| + |K_k| \cdot c_k) = \mathcal{O}(|K_k| \cdot |S_k|) = \mathcal{O}(|K_k| \cdot |E'_k|)$ time.

We proceed as follows during any step i . First, we scan $N_k(u_i)$ and, for every $s \in N_k(u_i)$ such that $\mathbf{A}[s] = 0$, we set $\mathbf{A}[s] = i$. Furthermore, if we set $\mathbf{A}[s] = i$, then we add every $u' \in N_k(s)$ in \mathbf{Stack} (we use the auxiliary array $\mathbf{InStack}$ in order to avoid adding twice a vertex). Then, we consider each vertex in \mathbf{Stack} sequentially, until we emptied the stack. For every vertex u' considered, we scan the list $N_k(u')$. If $s \in N_k(u')$ is such that $\mathbf{A}[s] = i$ (equivalently, if $s \in N_{k,i}(u_i)$) then, we increment $\mathbf{Int}[u'][0]$ by one (intersection size with $N_{k,i}(u_i)$). Similarly, we increment $\mathbf{Sum}[u'][0]$ by $\beta_k(s)$. For the unique t such that $s \in S_k^t$, we also increment $\mathbf{Int}[u'][t]$ by one and $\mathbf{Sum}[u'][t]$ by $\beta_k(s)$, respectively. Finally, we apply our above formula in order to increment Ψ_k :

1. For every $s \in N_k(u_i)$, if $\mathbf{A}[s] = i$, then let t be the unique index such that $s \in S_k^t$. We increment Ψ_k by $\beta_k(s) \times (\mathbf{Sum}[u_i][0] - \mathbf{Sum}[u_i][t])$.
2. For every $s \in S_k$ such that $\mathbf{A}[s] = 0$, let also t be the unique index such that $s \in S_k^t$. We scan the list $N_k(s)$ in order to find some u' maximizing $\mathbf{Int}[u'][0] - \mathbf{Int}[u'][t]$. Then, we increment Ψ_k by $2\beta_k(s) \times (\mathbf{Sum}[u'][0] - \mathbf{Sum}[u'][t])$.

The whole step only takes $\mathcal{O}(|E'_k|)$ time because each adjacency list is scanned $\mathcal{O}(1)$ times. ◀

Although it is a truly subquadratic algorithm (in the size $n + m$ of the input), our algorithm does not perform better than the classic $\mathcal{O}(nm)$ -time algorithm for general graphs if $m = \Theta(n^2)$. This is in sharp contrast with our results in Sec. 4, for hereditary chordal subclasses, where all our algorithms run in $\tilde{\mathcal{O}}(n^a m)$ time, for some $a < 1$. It would be very interesting to improve the running time of Theorem 17, and to bring it much closer to the linear-time complexity of the diameter problem on this subclass of graphs.

6 Open problems

We left open whether there are there relevant graph classes where the complexity of the Wiener index and the diameter are different. In particular, are both problems subquadratic equivalent? Another interesting question is whether we could compute the Wiener index of Helly graphs in truly subquadratic time. In this paper, we only managed to find such algorithm for the subclasses of dually chordal graphs and chordal Helly graphs.

References

- 1 A. Backurs, L. Roditty, G. Segal, V. Vassilevska Williams, and N. Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 267–280, 2018.
- 2 L. Bénéteau, J. Chalopin, V. Chepoi, and Y. Vaxès. Medians in median graphs and their cube complexes in linear time. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 3 J. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. ORNL, 1993.
- 4 John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer-Verlag London, 2008.
- 5 E. Bonnet. 4 vs 7 sparse undirected unweighted Diameter is SETH-hard at time $n^{4/3}$. Technical report, arXiv, 2021. [arXiv:2101.02312](https://arxiv.org/abs/2101.02312).
- 6 E. Bonnet. Inapproximability of diameter in super-linear time: Beyond the 5/3 ratio. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPICs*, pages 17:1–17:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. [doi:10.4230/LIPICs.STACS.2021.17](https://doi.org/10.4230/LIPICs.STACS.2021.17).
- 7 Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, April 2016. [doi:10.1016/j.entcs.2016.03.005](https://doi.org/10.1016/j.entcs.2016.03.005).
- 8 Nicolas Bousquet, Aurélie Lagoutte, Zhentao Li, Aline Parreau, and Stéphan Thomassé. Identifying codes in hereditary classes of graphs and vc-dimension. *SIAM Journal on Discrete Mathematics*, 29(4):2047–2064, 2015.
- 9 A. Brandstädt, V. Chepoi, and F.F. Dragan. The algorithmic use of hypertree structure and maximum neighbourhood orderings. *Discrete Applied Mathematics*, 82(1-3):43–77, 1998.
- 10 A. Brandstädt, V. Chepoi, and F.F. Dragan. Tree-structured graphs. In *Handbook of Graph Theory, Combinatorial Optimization, and Algorithms*. CRC Press, 2016.
- 11 A. Brandstädt, F.F. Dragan, V. Chepoi, and V. Voloshin. Dually chordal graphs. *SIAM Journal on Discrete Mathematics*, 11(3):437–455, 1998.
- 12 K. Bringmann, T. Husfeldt, and M. Magnusson. Multivariate Analysis of Orthogonal Range Searching and Graph Distances. *Algorithmica*, pages 1–24, 2020.
- 13 P. Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9(3):205–212, 1974.
- 14 S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Transactions on Algorithms*, 15(2):21, 2018.
- 15 G. Chang and G. Nemhauser. The k -domination and k -stability problems on sun-free chordal graphs. *SIAM Journal on Algebraic Discrete Methods*, 5(3):332–345, 1984.
- 16 B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete & Computational Geometry*, 4(5):467–489, 1989.
- 17 D. Corneil, F. F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. *Discrete Applied Mathematics*, 113(2-3):143–166, 2001.

- 18 M. Dalirrooyfard and N. Wein. Tight Conditional Lower Bounds for Approximating Diameter in Directed Graphs. In *53rd Annual ACM Symposium on Theory of Computing (STOC)*, 2021. To appear.
- 19 P. Dankelmann. Computing the average distance of an interval graph. *Information Processing Letters*, 48(6):311–314, 1993.
- 20 Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2010. 4th edition. doi:10.1007/978-3-662-53622-3.
- 21 F.F. Dragan, G. Ducoffe, and H.M. Guarnera. Fast deterministic algorithms for computing all eccentricities in (hyperbolic) Helly graphs, 2021. To appear. Technical report available on arXiv (arXiv:2102.08349).
- 22 G. Ducoffe and Feodor F.F. Dragan. A story of diameter, radius, and (almost) Helly property. *Networks*, 77(3):435–453, 2021.
- 23 G. Ducoffe, M. Habib, and L. Viennot. Fast diameter computation within split graphs. In *COCOA*, pages 155–167. Springer, 2019.
- 24 G. Ducoffe, M. Habib, and L. Viennot. Diameter computation on H -minor free graphs and graphs of bounded (distance) VC-dimension. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1905–1922. SIAM, 2020.
- 25 Guillaume Ducoffe. Around the diameter of AT-free graphs. *CoRR*, abs/2010.15814, 2020. arXiv:2010.15814.
- 26 Guillaume Ducoffe. Optimal diameter computation within bounded clique-width graphs. *CoRR*, abs/2011.08448, 2020. arXiv:2011.08448.
- 27 David Eppstein and Joseph Wang. Fast approximation of centrality. *Journal of Graph Algorithms and Applications*, 8(1):39–45, 2004. doi:10.7155/jgaa.00081.
- 28 Jacob Evald and Søren Dahlgaard. Tight hardness results for distance and centrality problems in constant degree graphs. Technical Report arXiv:1609.08403, ArXiv, 2016.
- 29 S. Foldes and P.L. Hammer. Split graphs having Dilworth number two. *Canadian Journal of Mathematics*, 29(3):666–672, 1977.
- 30 F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.
- 31 A. Goldman. Optimal center location in simple networks. *Transportation science*, 5(2):212–221, 1971.
- 32 R. Impagliazzo and R. Paturi. On the complexity of k -SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
- 33 C. Jordan. Sur les assemblages de lignes. *J. Reine Angew. Math*, 70(185):81, 1869.
- 34 Dirk Koschützki, Katharina Anna Lehmann, Leon Peeters, Stefan Richter, Dagmar Tenfelde-Podehl, and Oliver Zlotowski. Centrality indices. In *Network Analysis*, pages 16–61. Springer, 2005. doi:10.1007/978-3-7091-0741-6_9.
- 35 P.S. Kumar and C.V. Madhavan. Minimal vertex separators of chordal graphs. *Discrete Applied Mathematics*, 89(1-3):155–168, 1998.
- 36 R. Li. Settling SETH vs. Approximate Sparse Directed Unweighted Diameter (up to (NU)NSETH). In *53rd Annual ACM Symposium on Theory of Computing (STOC)*, 2021. To appear.
- 37 M. Moscarini. Doubly chordal graphs, Steiner trees, and connected domination. *Networks*, 23(1):59–69, 1993.
- 38 S. Olariu. A simple linear-time algorithm for computing the center of an interval graph. *International Journal of Computer Mathematics*, 34(3-4):121–128, 1990.
- 39 L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, pages 515–524, 2013.
- 40 R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on computing*, 13(3):566–579, 1984.

44:16 On Computing the Maximum and Average Distances for Some Chordal-Like Graphs

- 41 J. R. Walter. *Representations of rigid cycle graphs*. PhD thesis, Wayne State University, Department of Mathematics, 1972.
- 42 H. Wiener. Structural determination of paraffin boiling points. *Journal of the American chemical society*, 69(1):17–20, 1947.
- 43 D. Willard. New data structures for orthogonal range queries. *SIAM Journal on Computing*, 14(1):232–253, 1985.

A Cubic Vertex-Kernel for Trivially Perfect Editing

Maël Dumas

Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, F-45067 Orléans, France

Anthony Perez

Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, F-45067 Orléans, France

Ioan Todinca

Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, F-45067 Orléans, France

Abstract

We consider the TRIVIALY PERFECT EDITING problem, where one is given an undirected graph $G = (V, E)$ and a parameter $k \in \mathbb{N}$ and seeks to *edit* (add or delete) at most k edges from G to obtain a trivially perfect graph. The related TRIVIALY PERFECT COMPLETION and TRIVIALY PERFECT DELETION problems are obtained by only allowing edge additions or edge deletions, respectively. Trivially perfect graphs are both chordal and cographs, and have applications related to the tree-depth width parameter and to social network analysis. All variants of the problem are known to be NP-complete [6, 29] and to admit so-called polynomial kernels [13, 23]. More precisely, the existence of an $O(k^3)$ vertex-kernel for TRIVIALY PERFECT COMPLETION was announced by Guo [23] but without a stand-alone proof. More recently, Drange and Pilipczuk [13] provided $O(k^7)$ vertex-kernels for these problems and left open the existence of cubic vertex-kernels. In this work, we answer positively to this question for all three variants of the problem.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Parameterized complexity, kernelization algorithms, graph modification, trivially perfect graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.45

Related Version *Full Version*: <https://arxiv.org/abs/2105.08549>

Introduction

A broad range of optimization problems on graphs are particular cases of so-called modification problems. Given an arbitrary graph $G = (V, E)$ and an integer k , the question is whether G can be turned into a graph satisfying some desired property by at most k *modifications*. By modifications we mean, according to the problem, vertex deletions (as for VERTEX COVER and FEEDBACK VERTEX SET where we aim to obtain graphs with no edges, or without cycles respectively) or edge deletions and/or additions (as for MINIMUM FILL-IN, also known as CHORDAL COMPLETION, where the goal is to obtain a chordal graph, with no induced cycles with four or more vertices, by adding at most k edges).

Here we consider edge modifications problems, that can be split in three categories, depending whether we allow only edge additions, only edge deletions, or both operations, in which case we speak of edge editing. Consider a family \mathcal{H} of graphs, called *obstructions*. In the \mathcal{H} -FREE EDITING problem we seek to edit at most k edges of G to obtain a graph that does not contain any obstruction from \mathcal{H} as an induced subgraph. One can similarly define \mathcal{H} -FREE COMPLETION and \mathcal{H} -FREE DELETION variants of this problem by only allowing the addition or deletion of edges, respectively. E.g., MINIMUM FILL-IN corresponds to \mathcal{H} -FREE COMPLETION, where \mathcal{H} is formed by all cycles with at least four vertices. For most families \mathcal{H} , all three versions are NP-complete, but thinking of k as of some suitably small quantity, they have been intensively studied in the framework of *parameterized complexity* (see [11] for



© Maël Dumas, Anthony Perez, and Ioan Todinca;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 45; pp. 45:1–45:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a comprehensive survey). The aim of parameterized complexity is to determine whether it is possible to decide the instance at hand in time $f(k) \cdot n^{O(1)}$ for some computable function f . Such problems are said to be FPT (*fixed-parameter tractable*). With a simple but elegant and powerful argument, Cai [7] proved that whenever \mathcal{H} is finite all three variants are FPT. Basically, whenever the graph contains one of the obstructions (graphs of \mathcal{H}), the algorithm branches on all possible modifications to destroy it, and makes the recursive calls with a lesser parameter k . When the family \mathcal{H} contains all cycles with at least four vertices, the corresponding edition problem CHORDAL EDITING was shown to be FPT relatively recently [10]. The completion variant, i.e., the MINIMUM FILL-IN, was known to be FPT since the 90's [7, 25].

We consider an equivalent definition of fixed-parameter tractability, namely *kernelization*. Given a parameterized problem Π , a *kernelization algorithm* for Π (or *kernel* for short) is an algorithm that given any instance (I, k) of Π runs in time polynomial in $|I|$ and k and outputs an equivalent instance (I', k') of Π such that $|I'| \leq h(k)$ and $k' \leq g(k)$ for some computable functions g and h . Whenever h is polynomial, we say that Π admits a *polynomial kernel*. A kernelization algorithm uses a set of polynomial-time computable *reduction rules* to *reduce* the instance at hand. We say that a reduction rule is *safe* whenever its application on an instance (I, k) of Π results in an equivalent instance (I', k') of Π . It is well-known that a parameterized problem is FPT if and only if it admits a kernelization algorithm [17]. While many polynomial kernels are known to exist for editing problems (see [11] or [27] for surveys), it is known that some editing problems are unlikely to admit polynomial kernels under reasonable complexity-theoretic assumptions [8, 22, 26]. When \mathcal{H} contains only a single obstruction, several results towards a dichotomy regarding the existence of polynomial kernels have been obtained [1, 8, 28]. Very recently, Marx and Sandeep [28] narrowed down the problem for obstructions containing at least 5 vertices to only nine distinct obstructions. In other words, the non-existence of polynomial kernels for \mathcal{H} -FREE EDITING for all such obstructions would imply the non-existence of polynomial kernels for any obstruction with at least 5 vertices. When \mathcal{H} contains several obstructions, a very natural setting is to include all cycles of length at least 3 in \mathcal{H} , thus targeting a subclass of chordal graphs. Indeed, editing (and especially completion) problems towards such classes cover classical problems with both theoretical and practical interest [15, 21, 24, 25, 33]. Notice that many known polynomial kernels for editing problems concern such classes [3, 4, 13, 23, 25]. For completion and deletion versions, polynomial kernels are often used as a first step in the design of subexponential parameterized algorithms [5, 12, 18, 19].

In this work, we focus on editing problems towards trivially perfect graphs, that is $\mathcal{H} = \{P_4, C_4\}$ (respectively a path and a cycle on 4 vertices). This problem is known as TRIVIALY PERFECT EDITING in the literature. By allowing edge addition or edge deletion only, we obtain the TRIVIALY PERFECT COMPLETION and TRIVIALY PERFECT DELETION problems, respectively.

Related work

While the NP-Completeness of TRIVIALY PERFECT COMPLETION and TRIVIALY PERFECT DELETION has been known for some time [6], the complexity of TRIVIALY PERFECT EDITING remained open until a work of Nastos and Gao [29]. Trivially perfect graphs have recently regained attention since they are related to the well-studied width parameter *tree-depth* [20, 30] which corresponds to the size of the largest clique of a trivially perfect supergraph of G with the smallest clique number. Moreover, Nastos and Gao [29] proposed a new definition for

community structure based on small obstructions. In particular, the authors emphasized that editing a given graph into a trivially perfect graph *yields meaningful clusterings in real networks* [29]. Trivially perfect graphs also correspond to chordal cographs and admit a so-called *universal clique decomposition* [12]. Polynomial kernels with $O(k^7)$ vertices have been obtained for all variants of the problem by Drange and Pilipczuk [13]. The technique used relies on a reduction rule bounding the number of vertices in any trivially perfect *module* and the computation of a so-called *vertex modulator*, that is a maximal packing of obstructions with additional properties. Combined with sunflower-like reduction rules and a careful analysis of the graph remaining apart from the vertex modulator, the authors managed to provide polynomial kernels. They then asked whether the $O(k^7)$ bound could be improved, and qualify as “really challenging question” whether one can match the $O(k^3)$ bound for TRIVIALLY PERFECT COMPLETION claimed by Guo [23].

Our contribution

We answer positively to this question and provide kernels with $O(k^3)$ vertices for all considered problems. To be complete, a quadratic vertex-kernel for the completion version only is claimed in [2, 9]. While our kernelization algorithm shares similarities with the work of Drange and Pilipczuk [13], our technique differs in several points. In particular, we do not rely on the computation of a vertex modulator, a useful technique to design polynomial kernels but somehow responsible for the large bound obtained. To circumvent this issue, we only rely on the so-called universal clique decomposition of trivially perfect graphs. This decomposition partitions the vertices of trivially perfect graph G into cliques, the bags being structured as nodes of a rooted forest such that two vertices are adjacent in G if and only they are in a same bag, or in two bags such that one is an ancestor of the other in the forest. For any positive instance of the problem, at most $2k$ bags contain vertices incident to modified edges. Informally, the rest of the bags can be regrouped into two types of ‘chunks’. Some correspond to trivially perfect modules of the input graph (which are known to be reducible to small sizes by [13]), others have a more complicated but still particular structure, similar to the *combs* of [13]. We show how to reduce the size of these combs. Altogether we believe that our rules not only improve the size of the kernel but also significantly simplify the kernelization algorithm of [13]. Last but not least, we think that this approach based on tree-like decompositions and the analysis of large chunks of the graph that are not affected by the modified edges might be exploitable for other editing problems. Indeed the technique has strong similarities with the notion of branches introduced by Bessy et al. [3] for modification to 3-leaf power graphs, a closely related graph class.

Outline

We begin with some preliminaries definitions and results about trivially perfect graphs (Section 1). We then introduce the notion of combs and provide the set of reduction rules needed to obtain an $O(k^3)$ vertex-kernel for TRIVIALLY PERFECT EDITING (Section 2). The combinatorial bound on the kernel size is provided in Section 3. We explain how these results can be adapted to obtain similar kernels for TRIVIALLY PERFECT COMPLETION and TRIVIALLY PERFECT DELETION (Section 4). The Conclusion section summarizes the results and suggests further developments. Proofs of statements labeled with (\star) are omitted in this extended abstract. The interested reader may refer to [14] for a full version of this paper.

1 Preliminaries

We consider simple, undirected graphs $G = (V, E)$ where V denotes the *vertex set* and $E \subseteq (V \times V)$ the *edge set* of G . We will sometimes use $V(G)$ and $E(G)$ to clarify the context. Given a vertex $u \in V$, the *open neighborhood* of u is the set $N_G(u) = \{v \in V : uv \in E\}$. The *closed neighborhood* of u is defined as $N_G[u] = N_G(u) \cup \{u\}$. A vertex $u \in V$ is *universal* if $N_G[u] = V$, and two vertices u and v are *true twins* if $N_G[u] = N_G[v]$. The set of universal vertices forms a clique and is called the *universal clique* of G . Given a subset of vertices $S \subseteq V$, $N_G[S]$ is the set $\cup_{v \in S} N_G[v]$ and $N_G(S)$ is the set $N_G[S] \setminus S$. We will omit the mention to G whenever the context is clear. The subgraph *induced* by S is defined as $G[S] = (S, E_S)$ where $E_S = \{uv \in E : u \in S, v \in S\}$. For the sake of readability, given a subset $S \subseteq V$ we define $G \setminus S$ as $G[V \setminus S]$. A subset of vertices $C \subseteq V$ is a *connected component* of G if $G[C]$ is a maximal connected subgraph of G . A subset of vertices $M \subseteq V$ is a *module* of G if and only if $N_G(u) \setminus M = N_G(v) \setminus M$ holds for every $u, v \in M$. A maximal set of true twins $K \subseteq V$ is a *critical clique*. Notice that $G[K]$ is a clique module and that the set $\mathcal{K}(G)$ of critical cliques of any graph G partitions its vertex set $V(G)$. Notice that the universal clique is a critical clique.

Trivially perfect graphs

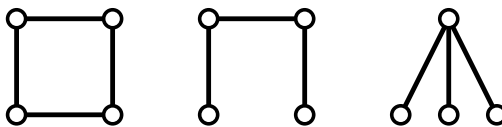
A graph $G = (V, E)$ is trivially perfect if and only if it does not contain any P_4 (a path on 4 vertices) nor C_4 (a cycle on 4 vertices) as an induced subgraph (see Figure 1). We consider the following problem.

TRIVIALY PERFECT EDITING

Input: A graph $G = (V, E)$, a parameter $k \in \mathbb{N}$

Question: Does there exist a set of pairs $F \subseteq (V \times V)$ of size at most k such that the graph $H = (V, E \Delta F)$ is trivially perfect, with $E \Delta F = (E \setminus F) \cup (F \setminus E)$?

Given an instance $(G = (V, E), k)$ of TRIVIALY PERFECT EDITING, a set $F \subseteq (V \times V)$ such that $H = (V, E \Delta F)$ is trivially perfect is an *edition* of G . When F is constrained to be disjoint from (resp. contained in) E , we say that F is a *completion* (resp. a *deletion*) of G . The corresponding problems are TRIVIALY PERFECT COMPLETION and TRIVIALY PERFECT DELETION, respectively. For the sake of simplicity, given an edition (resp. completion, deletion) F of G , we use $G \Delta F$, $G + F$ and $G - F$ to denote the graphs $(V, E \Delta F)$, $(V, E \cup F)$ and $(V, E \setminus F)$, respectively. A vertex is *affected* by F whenever it is contained in some pair of F . The set F is a *k-edition* (resp. *k-completion*, *k-deletion*) whenever $|F| \leq k$. Finally, we say that such a set F is *optimal* whenever it is minimum-sized.



■ **Figure 1** The C_4 , P_4 and claw graphs, respectively. The claw will be useful in some of our proofs.

Trivially perfect graphs are hereditary and closed under true twin addition. This property will be useful to deal with critical cliques, as stated by the following result. Recall that critical cliques are maximal sets of true twins (or, equivalently, maximal clique modules), they will play a central role throughout this paper.

► **Lemma 1** ([3]). *Let \mathcal{G} be a hereditary class of graphs closed under true twin addition. For every graph $G = (V, E)$, there exists an optimal edition (resp. completion, deletion) F into a graph of \mathcal{G} such that for any two critical cliques K and K' either $(K \times K') \subseteq F$ or $(K \times K') \cap F = \emptyset$.*

Several characterizations are known to exist for trivially perfect graphs. We will mainly use the following ones.

► **Proposition 2** ([32]). *The class of trivially perfect graphs can be defined recursively as follows:*

- a single vertex is a trivially perfect graph.
- Adding a universal vertex to a trivially perfect graph results in a trivially perfect graph.
- The disjoint union of two trivially perfect graphs results in a trivially perfect graph.

► **Definition 3** (Universal clique decomposition, [12]). *A universal clique decomposition (UCD) of a connected graph $G = (V, E)$ is a pair $\mathcal{T} = (T = (V_T, E_T), \mathcal{B} = \{B_t\}_{t \in V_T})$ where T is a rooted tree and \mathcal{B} is a partition of the vertex set V into disjoint nonempty subsets, such that:*

- if $vw \in E$ and $v \in B_t, w \in B_s$ then s and t are on a path from a leaf to the root, with possibly $s = t$, and
- for every node $t \in V_T$, the set of vertices B_t is the universal clique of the induced subgraph $G[\bigcup_{s \in V(T_t)} B_s]$, where T_t denotes the subtree of T rooted at t .

The vertices of T are called *nodes* of the decomposition, while the sets of \mathcal{B} are called *bags*. We will sometimes abuse notation and identify nodes of T with their corresponding bags in \mathcal{B} . Notice moreover that in a universal clique decomposition, every node t of T that is not a leaf has at least two children since otherwise B_t would not contain *all* universal vertices of $G[\bigcup_{s \in V(T_t)} B_s]$.

► **Lemma 4** ([12]). *A connected graph G admits a universal clique decomposition if and only if it is trivially perfect. Moreover, such a decomposition is unique up to isomorphisms.*

One can observe that finding a universal clique decomposition can be done in polynomial time by iteratively identifying universal cliques and connected components. Finally, both Definition 3 and Lemma 4 can be naturally extended to *disconnected* trivially perfect graphs by considering a *rooted forest* instead of a rooted tree. More precisely, the universal clique decomposition of a disconnected graph $G = (V, E)$ is a rooted forest of universal clique decompositions of its connected components. Such a graph is thus trivially perfect if and only if it admits a universal clique decomposition shaped like a rooted forest.

We conclude this section by providing a new characterization of trivially perfect graphs in terms of maximal cliques and nested families.

► **Definition 5** (Nested family). *Let U be a universe and $\mathcal{F} \subseteq 2^U$ a family of subsets of U . The family \mathcal{F} is nested if and only if for every $A, B \in \mathcal{F}$, $A \subseteq B$ or $B \subseteq A$ holds.*

► **Lemma 6** (*). *Let $G = (V, E)$ be a graph, $S \subseteq V$ a maximal clique of G and K_1, \dots, K_r the connected components of $G \setminus S$. The graph G is trivially perfect if and only if the following conditions are verified:*

- (i) $G[S \cup K_i]$ is trivially perfect for every $1 \leq i \leq r$,
- (ii) $\bigcup_{1 \leq i \leq r} \{N_G(K_i)\}$ is a nested family,
- (iii) $(K_i \times N_G(K_i)) \subseteq E$ for every $1 \leq i \leq r$.

2 Kernelization algorithm for Trivially Perfect Editing

We begin this section by providing a high-level description of our kernelization algorithm. As mentioned in the introductory section, we use the universal clique decomposition of trivially perfect graphs to bound the number of vertices of a reduced instance. Let us consider a positive instance $(G = (V, E), k)$ of TRIVIALY PERFECT EDITING, F a suitable solution and $H = G \Delta F$. Denote by $\mathcal{T} = (T, \mathcal{B})$ the universal clique decomposition of H as described Definition 3. Since $|F| \leq k$, we know that at most $2k$ bags of \mathcal{T} may contain affected vertices. Let A be the set of such bags, and let A' denote the lowest common ancestor closure of A in forest T (Definition 17). As we shall see later, the size of A' is also linear in k (Lemma 18). The removal of every bag of A' from T will disconnect the forest T into several components (see Figure 2).

Such a connected component D of $T \setminus A'$ may see zero, one or two nodes of A' in the forest T (Lemma 4). If D has no neighbour in A' , the union of all bags of D corresponds to a connected component of H and of G , inducing a trivially perfect graph in G , and will be eliminated by a reduction rule. We shall see that the union of all components D_a of the second type, seeing a unique bag $a \in A'$ in the forest T , corresponds to a trivially perfect module of graph G . We use the reductions rules of [13] to shrink such a module to $O(k^2)$ vertices, which boils down to a total $O(k^3)$ vertices since $|A'| = O(k)$.

Our efforts will be focused on components D seeing two bags $a_1, a_2 \in A'$, one of them being ancestor of the other in forest T . We call such a structure D a *comb* (Definition 9 and Figure 2).

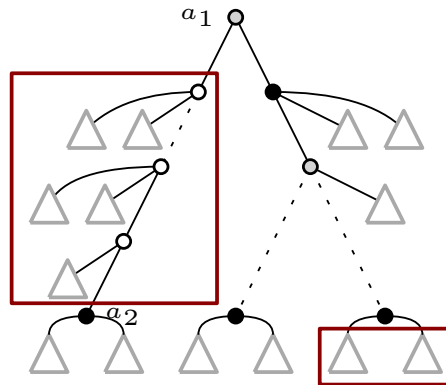


Figure 2 Analysis of a universal clique decomposition of a connected trivially perfect graph. Black vertices represent bags of A , gray vertices bags of A' and triangles are connected trivially perfect subgraphs of G . The leftmost rectangle is a comb of G , the rightmost a trivially perfect module. Note that any group of triangles rooted at a same bag is a trivially perfect module.

Such combs (the union of their bags) induce, in graph G , a trivially perfect subgraph that can be partitioned with regard to critical cliques and trivially perfect modules with nice inclusion properties on their neighborhoods. We provide two distinct reduction rules on these structures. Rule 4 reduces the so-called shaft of the comb (intuitively, the path strictly between a_1 and a_2 in T) to length $O(k)$. Rule 5 reduces the size of the whole comb (the union of its bags) to $O(k^2)$. Altogether, the reduced instance cannot contain more than $O(k^3)$ vertices.

We would like to note that the combs considered in this work are similar to the ones defined by Drange and Pilipczuk [13] and thus named after them. However, the two structures are not strictly identical, in particular since they were originally defined with respect to a vertex modulator (i.e. a packing of obstructions), and thus their neighborhood towards the rest of the graph was structured differently.

In the remaining of this section we assume that we are given an instance $(G = (V, E), k)$ of TRIVIALY PERFECT EDITING.

2.1 Reducing critical cliques and trivially perfect modules

We first give a classical reduction rule when dealing with modification problems. This rule is trivially safe for trivially perfect graphs.

► **Rule 1.** *Let $C \subseteq V$ be a subset of vertices such that $G[C]$ is a trivially perfect connected component of G . Remove C from G .*

We now give known reduction rules that deal with critical cliques and trivially perfect modules. The safeness of Rule 2 comes from the fact that trivially perfect graphs are hereditary and closed under true twin addition combined with Lemma 1. The safeness and polynomial-time application of Rule 3 was proved by Drange and Pilipczuk [13]. We would like to mention that while the statement of their rule assumes the instance at hand to be reduced by classical *sunflower* rules, this is actually not needed to prove the safeness of the rule. Altogether, we have the following.

► **Rule 2.** *Let $K \subseteq V$ be a set of true twins of G such that $|K| > k + 1$. Remove $|K| - (k + 1)$ arbitrary vertices in K from G .*

► **Rule 3.** *Let $M \subseteq V$ be a module of G such that $G[M]$ is trivially perfect and M contains an independent set I of size at least $2k + 5$. Remove all vertices of $M \setminus I$ from G .*

► **Lemma 7** (Folklore, [3, 13]). *Rules 1 to 3 are safe and can be applied in polynomial time.*

Using a structural result on trivially perfect graphs where critical cliques and independent sets have bounded size, Drange and Pilipczuk [13] proved the following.

► **Lemma 8** ([13]). *Let $(G = (V, E), k)$ be an instance of TRIVIALY PERFECT EDITING reduced under Rules 2 and 3. Then for every module $M \subseteq V$ such that $G[M]$ is trivially perfect, $|M| = O(k^2)$.*

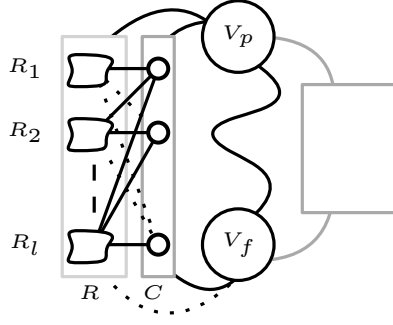
2.2 Reducing shafts of combs

We now consider the main structure of our kernelization algorithm, namely *combs*. Recall that such structures are similar to the ones defined by Drange and Pilipczuk [13] but not strictly identical. More precisely, the inner part of the structure is the same but not their neighborhoods towards the rest of the graph. We however choose to use the same name since it is well-suited to illustrate the structure (see Figure 3).

► **Definition 9** (Comb). *Let $G = (V, E)$ be a graph and $C, R \subseteq V$ be such that C is a clique which can be partitioned into l critical cliques $\{C_1, \dots, C_l\}$ and R can be partitioned into l non-empty and non-adjacent trivially perfect modules $\{R_1, \dots, R_l\}$. The pair $P = (C, R)$ is a comb if and only if:*

- *there exist $V_f, V_p \subseteq V(G) \setminus \{C, R\}$, $V_f \neq \emptyset$ such that $\forall x \in C$, $N_G(x) \setminus (C \cup R) = V_p \cup V_f$ and $\forall y \in R$, $N_G(y) \setminus (C \cup R) = V_p$,*
- *$N_G(C_i) \cap R = \bigcup_{j=i}^l R_j$ and $N_G(R_i) \cap C = \bigcup_{j=1}^i C_j$ for $1 \leq i \leq l$.*

By the following property, given a comb (C, R) of graph $G = (V, E)$, the subgraph $G[C \cup R]$ is trivially perfect, and has a universal clique decomposition in which critical cliques (C_1, \dots, C_l) are arranged in a path starting from the root, the *shaft* of the comb, and the decomposition of



■ **Figure 3** Illustration of a comb, with shaft C and teeth R . The edges between V_p and V_f can be anything. Every tooth R_i induces a (possibly disconnected) trivially perfect module.

each *tooth* R_i is attached to C_i ; see Figure 3. The length of (C, R) is l , the number of critical cliques in C . We can observe that $N_G[C_i] \subsetneq \cdots \subsetneq N_G[C_1]$ and $N_G(R_1) \subsetneq \cdots \subsetneq N_G(R_l)$ because for $1 \leq i \leq l$, $N_G[C_i] = (\bigcup_{j=i}^l R_j) \cup V_p \cup V_f$ and $N_G(R_i) = (\bigcup_{j=1}^i C_j) \cup V_p$.

► **Proposition 10** (\star). *Given a comb (C, R) of graph $G = (V, E)$, the subgraph $G[C \cup R]$ is trivially perfect. Moreover the sets V_p and V_f , and the ordered partitions (C_1, \dots, C_l) of C and (R_1, \dots, R_l) of R are uniquely determined.*

► **Lemma 11**. *Given an instance $(G = (V, E), k)$ of TRIVIALY PERFECT EDITING and a comb (C, R) of length $l \geq 2k + 2$ of G , there is no optimal k -edition that affects vertices in $C \cup R$.*

Proof. Consider a k -edition F of G and $H = G \Delta F$. Denote by $F' \subseteq F$ the subset of pairs from F which does not contain any vertex from $C \cup R$ and let $H' = G \Delta F'$. Since $|F| \leq k$ and (C, R) is a comb of length at least $2k + 2$, there exist $i \neq j \in \{1, \dots, l\}$ such that C_i, R_i, C_j and R_j do not include affected vertices of F . Let us take $c_1 \in C_i, r_1 \in R_i, c_2 \in C_j$ and $r_2 \in R_j$.

Suppose that H' is not trivially perfect, then there exists an obstruction W of H' such that $A = W \cap (C \cup R) \neq \emptyset$. Since pairs of F' do not contain vertices of $C \cup R$, (C, R) is a comb in H' and $|A| = 4$ is impossible since $H'[C \cup R] = G[C \cup R]$ is trivially perfect by Proposition 10. We show that $|A| = 3$ is also impossible. If $|A| = 3$ then the vertex $x \in W \setminus (C \cup R)$ is in the set $V_p \cup V_f$, otherwise the obstruction W would not be connected. We now show that $H'[W]$ is a claw, contains a triangle (as subgraph) or is not connected. If $x \in V_p$, then by construction x is adjacent to every vertex of the comb and $H'[W]$ would be a claw. If $x \in V_f$ and A contains at least two vertices in C , then these vertices would induce a triangle with x . If $x \in V_f$ and A contains at least two vertices $r', r'' \in R$, then x is not adjacent to any of them (since V_f does not see R in G). If r' and r'' are not adjacent in H' , either the fourth vertex of W sees r', r'' and x so $H'[W]$ is a claw, or $H'[W]$ is disconnected. If r' and r'' are adjacent in H' , they must belong to a same module R_i . Again the fourth vertex of W must either see them both thus forming a triangle, or none of them and $H'[W]$ is disconnected. In any case, A cannot be an obstruction and we conclude that either $|A| = 1$ or $|A| = 2$. We shall now construct an obstruction $W' = (W \setminus A) \cup A'$ such that $H'[W]$ and $H'[W']$ are isomorphic and $A' \subseteq \{c_1, r_1, c_2, r_2\}$. We can observe that W must contain a vertex from V_p or V_f .

■ If $|A| = 1$, take $x \in A$. If $x \in R$ then let $A' = \{r_1\}$, else let $A' = \{c_1\}$. Since (C, R) is a comb, $H'[W]$ and $H'[W']$ are isomorphic.

- If $|A| = 2$, denote by x and y the elements of A . If $x, y \in C$, then $H[W]$ contains a triangle. If $x \in C$ and $y \in R$, in the subcase $xy \in E(H')$ let $A' = \{c_1, r_1\}$ and observe that $c_1 r_1 \in E(H')$, hence $H'[W]$ and $H'[W']$ are isomorphic; in the other subcase $xy \notin E(H')$, take $A' = \{c_2, r_1\}$, so $c_2 r_1 \notin E(H')$ thus again $H'[W]$ and $H'[W']$ are isomorphic. Eventually consider the last case $x, y \in R$. If $xy \in E(H')$ then $H[W]$ contains a triangle, else $xy \notin E(H')$, so let $A' = \{r_2, r_1\}$ and note that $r_2 r_1 \notin E(H')$ thus $H'[W]$ and $H'[W']$ are isomorphic.

The set W' is an obstruction of H' and since the vertices in $\{c_1, r_1, c_2, r_2\}$ are not incident to any pair of F , W' is also an obstruction of H . Therefore H is not trivially perfect, which is a contradiction, concluding the proof of the Lemma. ◀

► **Rule 4.** *Given a comb (C, R) of length $l \geq 2k + 2$ of G , remove from G the vertices in $C_i \cup R_i$ for $2k + 2 < i \leq l$.*

► **Lemma 12** (*). *Rule 4 is safe.*

2.3 Reducing the teeth

► **Lemma 13.** *Let $(G = (V, E), k)$ be a yes-instance of TRIVIAALLY PERFECT EDITING, and (C, R) be a comb of G such that there exist $a, b \in \{1, \dots, l\}$ with $\sum_{a \leq i \leq l} |R_i| \geq 2k + 1$ and $\sum_{b \leq i < a} |R_i| \geq 2k + 1$. Then there exists an optimal k -edition F of G such that for every $m \in \{1, \dots, b - 1\}$, the vertices of R_m are all adjacent to the same vertices of $V(G) \setminus R_m$ in $G \Delta F$, and F contains no pair of vertices of R_m .*

Proof. Let F be an optimal k -edition of G and $H = G \Delta F$. There exist $v_2 \in (R_a \cup R_{a+1} \cup \dots \cup R_l)$ and $v_1 \in (R_b \cup R_{b+1} \cup \dots \cup R_{a-1})$ unaffected by F . The neighborhood of v_1 in $H \setminus R$ must be a clique: indeed, if there exist $x, y \in N_G(v_1) \setminus R$ such that $xy \notin E(H)$, then since $(N_G(v_1) \setminus R) \subseteq (N_G(v_2) \setminus R)$, the vertices $\{v_1, x, v_2, y\}$ would induce a C_4 . Let $1 \leq m < b$, we will construct an edition F_m such that $|F_m| \leq |F|$, F_m contains no pair of vertices included in R_m and the vertices of R_m are all adjacent to the same vertices in $G \Delta F_m$. Applying this construction iteratively to each R_m , $1 \leq m < b$ will yield an edition F^* that verifies the desired properties.

Let S be a maximal clique in H that contains $N_G(v_1) \setminus R$ and v_1 , and let K_1, \dots, K_r be the connected components of $H \setminus S$. Observe that K_1, \dots, K_r respect the conditions i, ii and iii of Lemma 6 with S . Let $v_m \in R_m$ be a vertex contained in the least number of pairs of F with the other element in S .

Denote by N the set of vertices of S adjacent to v_m in graph H . Let H' be the graph constructed from $H \setminus R_m$ and $G[R_m]$ by adding the edges $N \times R_m$, and F_m be the edition such that $H' = G \Delta F_m$. By construction $|F_m| \leq |F|$, we will now show that H' is trivially perfect.

We can observe that $R_m \cap S = \emptyset$ (because v_1 is unaffected by F and is non-adjacent with R_m in G) and therefore that S is a maximal clique of $H \setminus R_m$.

By construction of H' , S is also a maximal clique of H' and R_m is a connected component of $H_m \setminus S$. Let $K'_1, \dots, K'_{r'}$ be the connected components of $(H \setminus R_m) \setminus S$. Sets $K'_1, \dots, K'_{r'}$ verify the conditions i, ii and iii of Lemma 6 with respect to S in $H \setminus R_m$ and thus also in H' . Moreover $H'[S \cup R_m]$ is trivially perfect and $(N_{H'}(R_m) \times R_m) \subseteq E(H')$ by construction. The family $\bigcup_{1 \leq i \leq r} \{N_H(K_i)\}$ is nested according to Lemma 6, and, by construction of H' , $\bigcup_{1 \leq i \leq r'} \{N_{H'}(K'_i)\} \subseteq \bigcup_{1 \leq i \leq r} \{N_H(K_i)\}$. We also have that $N \in \bigcup_{1 \leq i \leq r} \{N_H(K_i)\}$. Indeed, let $\bar{K}(v_m)$ the connected component of $H \setminus S$ containing v_m , according to condition iii from Lemma 6 we have $N_H(\bar{K}(v_m)) = N_H(v_m) \cap S = N$. Therefore the family $\bigcup_{1 \leq i \leq r'} \{N_{H'}(K'_i)\} \cup \{N\}$ is also nested. By Lemma 6 applied on H' and S , graph H' is trivially perfect.

45:10 A Cubic Vertex-Kernel for Trivially Perfect Editing

As mentioned previously, we can apply this construction iteratively to each R_m , $1 \leq m < b$ and obtain an edition F^* that verifies the desired properties. \blacktriangleleft

► **Rule 5.** *Given a comb (C, R) of G such that there exist $a, b \in \{1, \dots, l\}$ with $\sum_{a \leq i \leq l} |R_i| \geq 2k + 1$ and $\sum_{b \leq i < a} |R_i| \geq 2k + 1$. Then for every $i \in \{1, \dots, b - 1\}$, replace R_i by a clique of size $\min(|R_i|, k + 1)$ with the same neighborhood.*

► **Lemma 14** (\star). *Rule 5 is safe.*

► **Lemma 15** (\star). *Let $(G = (V, E), k)$ be an instance of TRIVIALY PERFECT EDITING such that Rules 2 to 5 are not applicable. Then, for every comb (C, R) of G , $|C \cup R| = O(k^2)$.*

► **Lemma 16** (\star). *Given an instance $(G = (V, E), k)$ of TRIVIALY PERFECT EDITING, Rules 4 and 5 can be exhaustively applied in polynomial time.*

3 Bounding the size of a reduced instance

We now prove thoroughly that any reduced yes-instance of TRIVIALY PERFECT EDITING contains $O(k^3)$ vertices. To that end, we need the following definition and result.

► **Definition 17** (LCA-closure [16]). *Let $T = (V, E)$ be a rooted tree and $A \subseteq V(T)$. The lowest common ancestor-closure (LCA-closure) A' of A is obtained as follows. Initially, set $A' = A$. Then, as long as there exist $x, y \in A'$ whose lowest common ancestor w is not in A' , add w to A' . The LCA-closure of A is the last set A' obtained using this process.*

► **Lemma 18** ([16]). *Let $T = (V, E)$ be a rooted tree, $A \subseteq V(T)$ and $A' = \text{LCA-closure}(A)$. Then $|A'| \leq 2 \cdot |A|$ and for every connected component C of $T \setminus A'$, $|N_T(C)| \leq 2$.*

► **Theorem 19.** *TRIVIALY PERFECT EDITING admits a kernel with $O(k^3)$ vertices.*

Proof. Let $(G = (V, E), k)$ be a reduced yes-instance of TRIVIALY PERFECT EDITING and F a k -edition of G . Let $H = G \triangle F$ and $\mathcal{T} = (T, \mathcal{B})$ the universal clique decomposition of H . The graph G is not necessarily connected, thus T is a forest. Let A be the set of nodes $t \in V(T)$ such that the bag B_t contains a vertex affected by F . Since $|F| \leq k$, we have $|A| \leq 2k$. Let $A' \subseteq V(T)$ be the set containing the nodes of $\text{LCA-closure}(A)$ and the root of each connected component of T (in case the closure does not contain them). According to Lemma 18 and Rule 1 which implies that there are at most $2k$ connected components in G and thus $2k$ roots, we have $|A'| \leq 6k$.

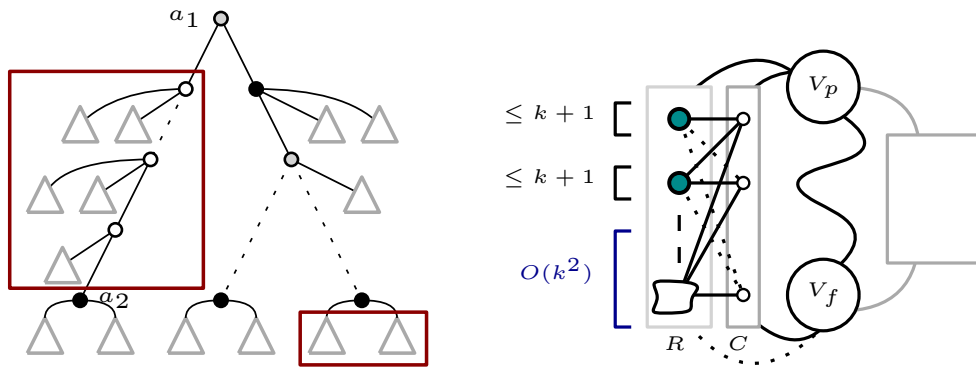
Let D be a connected component of $T \setminus A'$. We can observe that, by construction of A' (which for every pair of nodes, contains also the smallest common ancestor in T), only three cases are possibles (see Figure 4):

- $N_T(D) = \emptyset$ (D is a connected component of T).
- $N_T(D) = \{a\}$ (D is a subtree of T whose parent is $a \in A'$).
- $N_T(D) = \{a_1, a_2\}$ with one of the nodes $a_1, a_2 \in A'$ being an ancestor of the other in T .

We will say that these connected components are respectively of type 0, 1 or 2. For $D \subseteq V(T)$, we denote by $W(D) = \bigcup_{t \in D} B_t$ the set of vertices of G corresponding to bags of D .

There is no connected component of type 0 or else $W(D)$ would be a connected component of G inducing a trivially perfect graph. Rule 1 would have been applied to this component, contradicting the fact that G is a reduced instance.

Now consider the set of type 1 components D_1, D_2, \dots, D_r of $T \setminus A'$ attached in T to the same node $a \in A'$. We show that $W_a = W(D_1) \cup W(D_2) \cup \dots \cup W(D_r)$ is a trivially perfect module of G . In the graph H , W_a is by construction a module of the decomposition. Since



■ **Figure 4** (Left) universal clique decomposition of a connected component of H . (Right) shape of a reduced comb.

no vertex of W_a was affected by the edition F , W_a is also a module of G , trivially perfect by heredity. By Lemma 8, we have $|W_a| = O(k^2)$. There are at most $|A'| \leq 6k$ such sets W_a , thus the set of vertices of G in bags of type 1 components is of size $O(k^3)$.

Now consider the type 2 connected components D of $T \setminus A'$ which have two neighbors in T . Let a_1 and a_2 be these neighbors, one being the ancestor of the other, say a_1 is the ancestor of a_2 . Let t_1, \dots, t_l be the nodes of the tree on the path from a_1 to a_2 , in this order. The component D can be seen as a comb of shaft $(B_{t_1}, \dots, B_{t_l})$. More precisely, by construction of the universal clique decomposition, $W(D)$ can be partitioned into a comb (C, R) of H : the critical clique decomposition of C is $(C_1 = B_{t_1}, \dots, C_l = B_{t_l})$, and each R_i corresponds to the union of bags of the subtrees rooted at t_i which do not contain t_{i+1} , for $1 \leq i < l$, and to the union of bags of the subtrees rooted at t_l which do not contain a_2 , for $i = l$. Since (C, R) was not affected by F , it is also a comb of G . Thus for each type 2 component D , $W(D)$ contains $O(k^2)$ vertices by Lemma 15. Since T is a forest, it can contain at most $|A'| - 1 \leq 6k - 1$ such components in $T \setminus A$. Therefore the set of bags containing type 2 connected components of $T \setminus A$ contains $O(k^3)$ vertices.

It remains to bound the set of vertices of G which are in bags of A' . The vertices corresponding to nodes of $A' \setminus A$ are critical cliques of G , and are hence of size at most $k + 1$ by Rule 2. Thus the set of vertices in bags of $A' \setminus A$ is of size $O(k^2)$. The vertices corresponding to nodes of A are critical cliques in H but not necessarily of G . Let B_a be a bag corresponding to a node $a \in A$. We will show that B_a is covered by at most $2k + 1$ critical cliques of G , which by Rule 2 will imply that B_a contains $O(k^2)$ vertices of G , and thus the set of vertices in bags of A' is of size $O(k^3)$.

To see this, observe that B_a is a critical clique of H , and that G is obtained from H by editing at most k pairs of vertices. A result from [31] claims that, starting from a graph H and editing an edge, we add at most two critical cliques. The same arguments allow to claim that if B is a set of vertices covered by at most p critical cliques in H , and if H' is obtained by editing a pair of vertices x, y of H , then $p + 2$ critical cliques are enough to cover B in H' . To be complete, we now show this claim. Let $C_1, C_2, \dots, C_p, \dots, C_q$ be the critical cliques of H , suppose that B is covered by the first p cliques C_1, \dots, C_p . For each i , $1 \leq i \leq q$, the set $C''_i = C_i \setminus \{x, y\}$ is a clique module (not necessarily maximal) of H' . In particular, each C''_i is contained in a critical clique C'_i of H (the C'_i are not necessarily distinct). Let $C'(x)$ and $C'(y)$ be the critical cliques of H' containing respectively x and y . Clearly, the critical cliques $C'_1, \dots, C'_p, C'(x)$ and $C'(y)$ of H' cover the vertices of B , showing our claim. By applying this argument k times (one for each pair of F) to the bag B_a , which was a critical clique of H , we conclude that it is covered by at most $2k + 1$ critical cliques of G . Thus $|B_a| = O(k^2)$ by Rule 2.

45:12 A Cubic Vertex-Kernel for Trivially Perfect Editing

We conclude that $|V(G)| = O(k^3)$. Finally, we claim that a reduced instance can be computed in polynomial time. Indeed, Lemma 7 states that it is possible to reduce exhaustively a graph under Rules 2 to 3. Once this is done, it remains to apply exhaustively Rules 4 and 5 which is ensured by Lemma 16. ◀

4 Kernels for trivially perfect completion/deletion

In this section we show that the rules used for TRIVIALY PERFECT EDITING are safe for TRIVIALY PERFECT COMPLETION and TRIVIALY PERFECT DELETION. First Rules 1, 2 and 3 are safe for both problems. Indeed, the safeness of Rule 2 directly follows from Lemma 1 and Rule 3 was shown safe in [13].

We will now argue that Rules 4 and 5 are also safe. Lemma 11 states that no trivially perfect edition for an instance $(G = (V, E), k)$ of TRIVIALY PERFECT EDITING affects a comb of G of length at least $2k + 2$. This is also true when allowing only edge addition or edge deletion, implying the safeness of Rule 4 in both cases. In the proof of Lemma 13, for a trivially perfect edition F we construct another edition $F' \subseteq F$. In case F consists only of edge additions or deletions, it is also the case for F' , thus Lemma 13 holds for TRIVIALY PERFECT COMPLETION and TRIVIALY PERFECT DELETION and Rule 5 is safe for these problems.

The proof for the size of the kernel is the same as the proof of Theorem 19. Altogether, we obtain the following result.

► **Theorem 20.** *TRIVIALY PERFECT COMPLETION and TRIVIALY PERFECT DELETION admits a kernel with $O(k^3)$ vertices.*

5 Conclusion

We have provided a kernelization algorithm for TRIVIALY PERFECT EDITING, producing a cubic vertex-kernel, hence improving upon the $O(k^7)$ -size kernel of [13]. The techniques extend to the deletion and completion versions of the problem, within the same bounds. A natural question is whether the size of the kernel for TRIVIALY PERFECT EDITING can still be reduced – note that for TRIVIALY PERFECT COMPLETION, Bathie et al. claim a quadratic kernel [2].

Some ideas used in this work remind of very similar techniques applied to kernelization problems for edge editing towards classes of graphs \mathcal{G} having a tree-like decomposition. The simplest case – like here or for the class of so-called 3-leaf power graphs, see [3] – is when the vertices of the graph can be partitioned into bags inducing modules, and these bags can be structured as nodes of a forest T , with specific adjacency rules. If an arbitrary graph G can be turned into a graph of class \mathcal{G} by editing at most k pairs of vertices, the edited pairs are in some set A of at most $2k$ bags. Again by taking the lowest common ancestor closure A' of A , set A' is of size $O(k)$ and its removal from forest T will produce some chunks attached in T to 0, 1 or 2 nodes of A' (e.g., in [3], the authors speak of 1 and 2-branches, playing similar roles to modules and combs in this article). Kernelization algorithms can be obtained if we are able to reduce the bags themselves as well as the chunks, which hopefully have good structural properties. It is natural to wonder how general are these techniques, especially on subclasses of chordal graphs.

References

- 1 N. R. Aravind, R. B. Sandeep, and Naveen Sivadasan. Dichotomy results on the hardness of h-free edge modification problems. *SIAM J. Discret. Math.*, 31(1):542–561, 2017. doi:10.1137/16M1055797.
- 2 Gabriel Bathie, Nicolas Bousquet, and Théo Pierron. (Sub)linear kernels for edge modification problems towards structured graph classes. In preparation, 2021.
- 3 Stéphane Bessy, Christophe Paul, and Anthony Perez. Polynomial kernels for 3-leaf power graph modification problems. *Discret. Appl. Math.*, 158(16):1732–1744, 2010. doi:10.1016/j.dam.2010.07.002.
- 4 Stéphane Bessy and Anthony Perez. Polynomial kernels for proper interval completion and related problems. *Inf. Comput.*, 231:89–108, 2013. doi:10.1016/j.ic.2013.08.006.
- 5 Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michał Pilipczuk. A subexponential parameterized algorithm for proper interval completion. *SIAM J. Discret. Math.*, 29(4):1961–1987, 2015. doi:10.1137/140988565.
- 6 Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. Np-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006. doi:10.1016/j.dam.2006.03.031.
- 7 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.
- 8 Leizhen Cai and Yufei Cai. Incompressibility of H-free edge modification problems. *Algorithmica*, 71(3):731–757, 2015. doi:10.1007/s00453-014-9937-x.
- 9 Yixin Cao and Yuping Ke. Improved kernels for edge modification problems, 2021. arXiv:2104.14510.
- 10 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. doi:10.1007/s00453-015-0014-x.
- 11 Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. *CoRR*, abs/2001.06867, 2020. arXiv:2001.06867.
- 12 Pål Grønås Drange, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. Exploring the subexponential complexity of completion problems. *ACM Trans. Comput. Theory*, 7(4):14:1–14:38, 2015. doi:10.1145/2799640.
- 13 Pål Grønås Drange and Michał Pilipczuk. A polynomial kernel for trivially perfect editing. *Algorithmica*, 80(12):3481–3524, 2018. doi:10.1007/s00453-017-0401-6.
- 14 Maël Dumas, Anthony Perez, and Ioan Todinca. A cubic vertex-kernel for trivially perfect editing. *CoRR*, abs/2105.08549, 2021. arXiv:2105.08549.
- 15 Ehab S El-Mallah and Charles J Colbourn. The complexity of some edge deletion problems. *IEEE transactions on circuits and systems*, 35(3):354–362, 1988. doi:10.1109/31.1748.
- 16 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 470–479. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.62.
- 17 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.
- 18 Fedor V. Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM J. Comput.*, 42(6):2197–2216, 2013. doi:10.1137/11085390X.
- 19 Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015. doi:10.1007/s00453-013-9837-5.
- 20 Martin Charles Golumbic. Trivially perfect graphs. *Discret. Math.*, 24(1):105–107, 1978. doi:10.1016/0012-365X(78)90178-4.

- 21 Martin Charles Golumbic, Haim Kaplan, and Ron Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics*, 15(3):251–261, 1994. doi:10.1006/aama.1994.1009.
- 22 Sylvain Guillemot, Frédéric Havet, Christophe Paul, and Anthony Perez. On the (non-)existence of polynomial kernels for P_L -free edge modification problems. *Algorithmica*, 65(4):900–926, 2013. doi:10.1007/s00453-012-9619-5.
- 23 Jiong Guo. Problem kernels for NP-complete edge deletion problems: Split and related graphs. In Takeshi Tokuyama, editor, *Algorithms and Computation, 18th International Symposium, ISAAC 2007, Sendai, Japan, December 17-19, 2007, Proceedings*, volume 4835 of *Lecture Notes in Computer Science*, pages 915–926. Springer, 2007. doi:10.1007/978-3-540-77120-3_79.
- 24 Pavol Hell, Ron Shamir, and Roded Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM J. Comput.*, 31(1):289–305, 2001. doi:10.1137/S0097539700372216.
- 25 Haim Kaplan, Ron Shamir, and Robert Endre Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM J. Comput.*, 28(5):1906–1922, 1999. doi:10.1137/S0097539796303044.
- 26 Stefan Kratsch and Magnus Wahlström. Two edge modification problems without polynomial kernels. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 264–275. Springer, 2009. doi:10.1007/978-3-642-11269-0_22.
- 27 Yunlong Liu, Jianxin Wang, and Jiong Guo. An overview of kernelization algorithms for graph modification problems. *Tsinghua Science and Technology*, 19(4):346–357, 2014. doi:10.1109/TST.2014.6867517.
- 28 Dániel Marx and R. B. Sandeep. Incompressibility of h -free edge modification problems: Towards a dichotomy. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 72:1–72:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ESA.2020.72.
- 29 James Nastos and Yong Gao. Familial groups in social networks. *Soc. Networks*, 35(3):439–450, 2013. doi:10.1016/j.socnet.2013.05.001.
- 30 Jaroslav Nešetřil and Patrice Ossona de Mendez. On low tree-depth decompositions. *Graphs Comb.*, 31(6):1941–1963, 2015. doi:10.1007/s00373-015-1569-7.
- 31 Fábio Protti, Maise Dantas da Silva, and Jayme Luiz Szwarcfiter. Applying modular decomposition to parameterized cluster editing problems. *Theory Comput. Syst.*, 44(1):91–104, 2009. doi:10.1007/s00224-007-9032-7.
- 32 Jing-Ho Yan, Jer-Jeong Chen, and Gerard J Chang. Quasi-threshold graphs. *Discrete applied mathematics*, 69(3):247–255, 1996.
- 33 Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981. doi:10.1137/0602010.

Lower Bounds on Avoiding Thresholds

Robert Ferens  

Institute of Computer Science, University of Wrocław, Wrocław, Poland

Marek Szykuła  

Institute of Computer Science, University of Wrocław, Wrocław, Poland

Vojtěch Vorel 

Faculty of Informatics and Management, University of Hradec Králové, Czech Republic

Abstract

For a DFA, a word *avoids* a subset of states, if after reading that word the automaton cannot be in any state from the subset regardless of its initial state. A subset that admits an avoiding word is *avoidable*. The *k-avoiding threshold* of a DFA is the smallest number such that every avoidable subset of size k can be avoided with a word no longer than that number. We study the problem of determining the maximum possible k -avoiding thresholds. For every fixed $k \geq 1$, we show a general construction of strongly connected DFAs with n states and the k -avoiding threshold in $\Theta(n^k)$. This meets the known upper bound for $k \geq 3$. For $k = 1$ and $k = 2$, the known upper bounds are respectively in $\mathcal{O}(n^2)$ and in $\mathcal{O}(n^3)$. For $k = 1$, we show that $2n - 3$ is attainable for every number of states n in the class of strongly connected synchronizing binary DFAs, which is supposed to be the best possible in the class of all DFAs for $n \geq 8$. For $k = 2$, we show that the conjectured solution for $k = 1$ (an upper bound in $\mathcal{O}(n)$) also implies a tight upper bound in $\mathcal{O}(n^2)$ on 2-avoiding threshold. Finally, we discuss the possibility of using k -avoiding thresholds of synchronizing automata to improve upper bounds on the length of the shortest reset words.

2012 ACM Subject Classification Theory of computation \rightarrow Formal languages and automata theory; Mathematics of computing \rightarrow Discrete mathematics

Keywords and phrases avoiding word, Černý conjecture, rank conjecture, reset threshold, reset word, synchronizing automaton, synchronizing word

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.46

Funding *Robert Ferens*: Supported in part by the National Science Centre, Poland under project number 2017/25/B/ST6/01920.

Marek Szykuła: Supported in part by the National Science Centre, Poland under project number 2017/25/B/ST6/01920.

1 Introduction

A *deterministic finite complete semi-automaton* (called simply *automaton*) is a 3-tuple (Q, Σ, δ) , where Q is a finite set of *states*, Σ is an *input alphabet*, and $\delta: Q \times \Sigma \rightarrow Q$ is the *transition function*. The transition function is naturally extended to a function $Q \times \Sigma^* \rightarrow Q$. Throughout the paper, by n we always denote the number of states in Q .

Given a subset $S \subseteq Q$, the *image* of S under the action of a word $w \in \Sigma^*$ is $\delta(S, w) = \{\delta(q, w) \mid q \in S\}$. The *preimage* of S under the action of w is $\delta^{-1}(S, w) = \{q \in Q \mid \delta(q, w) \in S\}$.

The *rank* of a word w is the cardinality of the image $\delta(Q, w)$. A word w is *reset* if it has rank 1, i.e., its action maps all the states to one state. If an automaton admits a reset word, then it is called *synchronizing*, and its *reset threshold* is the length of the shortest reset word.

The central problem in the theory of synchronizing automata is the famous Černý conjecture, which states that every synchronizing n -state automaton has reset threshold at most $(n-1)^2$. Fig. 1 shows the 4-state automaton from the well-known Černý series [2], which



© Robert Ferens, Marek Szykuła, and Vojtěch Vorel;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

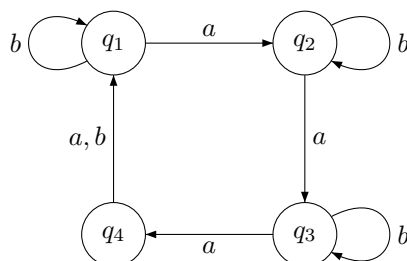
Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 46; pp. 46:1–46:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

meets the conjectured upper bound for every n . The Černý conjecture is one of the most longstanding open problems in automata theory, with constantly growing literature around the topic; see an old survey [15] and a recent special issue dedicated to the problem [17]. The best known general upper bound on reset threshold is cubic and equals $\sim 0.1654n^3 + o(n^3)$ [11]. Better bounds are known for many subclasses of automata.



■ **Figure 1** The Černý automaton with $n = 4$ states.

1.1 Avoiding words

Avoiding words are defined similarly to reset words. For a state $q \in Q$, a word w is *avoiding* if $q \notin \delta(Q, w)$, i.e., no state is mapped by the action of w to q . More generally, a word w *avoids* a subset $S \subset Q$ if $\delta(Q, w) \cap S = \emptyset$. Note that a word of rank $n - k$ is also a word avoiding some subset S of size k . In this way, a reset word is a specific case of an avoiding word.

A subset S is *avoidable* if there exists an avoiding word for S . Then, the *S -avoiding threshold* is the length of the shortest words avoiding S . The *k -avoiding threshold* is the maximum S -avoiding threshold over all subsets $S \subset Q$ of size k . In other words, every avoidable subset of size k can be avoided with a word of length not exceeding the k -avoiding threshold.

Obviously, a k -avoiding threshold is never larger than the $(k + 1)$ -avoiding threshold. In a synchronizing automaton, every subset of size $\leq n - 1$ is avoidable.

For example, for the automaton from Fig. 1, the k -avoiding thresholds for $k = 1, 2, 3$ are respectively equal to 4, 8, 12. For instance, the shortest word avoiding subset $\{q_2, q_3\}$ is ba^3ba^3 , and no other subset of size two requires a longer word.

Avoiding words are closely related to reset words and can be interesting as such for similar reasons. Yet, so far the focus was put on their application to bounding reset thresholds.

Originally, the concept was first used by Trahtman as a tool for improving the cubic upper bound on the reset threshold [14]. This turned out to be wrong as is based on the claim that 1-avoiding threshold is at most n , whereas it can be larger [5]. Nevertheless, the idea of applying avoiding words has been shown to be useful. A non-trivial quadratic upper bound on 1-avoiding threshold already led to the first improvement [12] of the old and well-known upper bound on the reset threshold [8]. This was later refined to $\sim 0.1654n^3 + o(n^3)$ [11], using the same method but improving the counting argument.

Better upper bounds on avoiding thresholds should lead to better upper bounds on reset thresholds. Yet, the problem of avoiding may be of similar difficulty.

Avoiding a subset is closely related to subset reachability. The latter is the question for a given subset $T \subseteq Q$, are there and how long are words w such that $\delta(Q, w) = T$. It is known that the decision problem is PSPACE-complete even if the automaton is strongly connected [18] and the shortest such words can have a length larger than $2^n/n$ [4]. It

holds similarly in the weaker *included* version where w is such that $\delta(Q, w) \subseteq T$ [4]. This is equivalent to avoiding the complement of T : $\delta(Q, w) \subseteq T$ is equivalent to $\delta(Q, w) \cap (Q \setminus T) = \emptyset$. It is also equivalent to that w is *totally extending*, i.e., $\delta^{-1}(T, w) = Q$, which was shown to be PSPACE-complete even if the automaton is strongly connected and binary [1]. It follows that avoiding a subset, in general, should require exponentially long words, but precise bounds were not shown so far. In particular, these results do not say what happens in the case of a synchronizing automaton nor if the size of the subset S to avoid is small. Yet, as we note, these cases are particularly important.

1.2 Known upper bounds

For the 1-avoiding threshold, the best known upper bound is quadratic in n . It is derived through linear algebraic methods applied to automata.

► **Theorem 1** (rephrased [12, Corollary 5]). *For an n -state automaton, the 1-avoiding threshold is at most $(n - 2)(n - 1) + 2$.*

In the general case, for k -avoiding threshold, we have the following asymptotic bound:

► **Theorem 2** (rephrased [1, Theorem 12]). *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be an n -state automaton, let r be the minimal rank in \mathcal{A} over all words, and let m be the length of the shortest words of the minimal rank. Then the k -avoiding threshold is at most $\mathcal{O}(n^{\min(r,k)} + m)$.*

Since for m we have only a cubic upper bound $\mathcal{O}(n^3)$, this component is dominating for $k = 1$ and $k = 2$. In these cases, it is unlikely to be tight. There is also the well-known rank (Pin-Černý) conjecture [8] stating that $m \leq (n - r)^2$.

No better bounds on avoiding thresholds are known in the case of a synchronizing and/or strongly connected automaton. Except for the obvious fact that for a synchronizing automaton $\mathcal{O}(n^3)$ is an upper bound on every k -avoiding threshold, and $\mathcal{O}(n^2)$ would be an upper bound if the (weak version of) Černý conjecture holds.

We note that it is also an easy exercise to prove that avoiding thresholds are small (at most linear) for many subclasses of automata. For example, for Eulerian automata, the k -avoiding threshold is at most $k(n - 1)$ [6]. For aperiodic automata [13, 16], or more generally, for automata with letters whose transitions contain only trivial cycles (self-loops), it is at most $n - 1$.

1.3 Known lower bounds

Concerning automata with the largest possible avoiding thresholds, only a few particular examples of automata were described so far. Moreover, they were limited to 1-avoiding threshold.

The first such example is a 4-state automaton with the 1-avoiding threshold equal to 6, which was found as a counterexample to the conjecture that the 1-avoiding threshold is bounded above by the number of states [5]. Later experiments [7] revealed several other examples with $n \leq 11$ states, suggesting that $2n - 2$ may be an upper bound.

It is also known that deciding whether 1-avoiding threshold or $\{q\}$ -avoiding threshold (for a given q) is smaller than a given integer is NP-complete [1].

1.4 Contribution

We tackle the problem of constructing automata with large avoiding thresholds. In this paper, we summarize our efforts.

First, we show that 1-avoiding threshold can be equal to $2n - 3$, for every number of states $n \geq 2$, and this is met by a series of strongly connected synchronizing automata. We conjecture that it is best possible, except for finitely many examples meeting $2n - 2$.

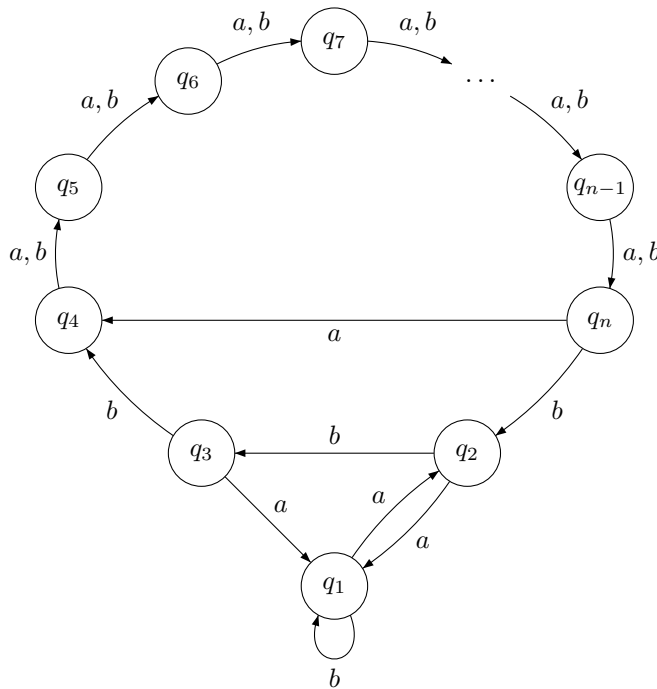
For the general case, we show a series of automata with k -avoiding thresholds in $\Theta(n^k)$. These automata are strongly connected but not synchronizing. This matches the asymptotic upper bound for $k \geq 3$, leaving open the cases $k = 2$ and $k = 1$. Yet, we show that the case of $k = 2$ can be reduced to a possible solution of the case of $k = 1$.

Finally, we note the potential of avoiding words for improving upper bounds on the reset threshold. So far, only an application of 1-avoiding threshold was considered; being in $\mathcal{O}(n)$, it would imply an upper bound on the reset threshold equal to $7/48n^3 + \mathcal{O}(n^2)$ [14]. However, with the generalized concept of avoiding subsets of size k , we can achieve more. If the k -avoiding threshold is subquadratic for all k bounded by any growing function in n (e.g., if for all $k \leq \log n$, the k -avoiding threshold is $o(n^2)$), then the reset threshold is subcubic.

2 1-avoiding threshold

We show a series of binary strongly connected synchronizing automata with 1-avoiding threshold equal to $2n - 3$. The existence of the series has been mentioned several times ([7, 12]), but it has not been described yet.

For each $n \geq 5$, we define $\mathcal{A}_n = (Q = \{q_1, \dots, q_n\}, \Sigma = \{a, b\}, \delta)$, which is shown in Fig. 2. The transition function is defined as follows: $\delta(q_1, a) = q_2; \delta(q_1, b) = q_1; \delta(q_2, a) = q_1; \delta(q_2, b) = q_3; \delta(q_3, a) = q_1; \delta(q_3, b) = q_4; \delta(q_i, a) = \delta(q_i, b) = q_{i+1}$ for all $i \in \{4, \dots, n-1\}; \delta(q_n, a) = q_4; \delta(q_n, b) = q_2$.



■ **Figure 2** The binary automaton \mathcal{A}_n with the 1-avoiding threshold equal to $2n - 3$.

► **Proposition 3.** *For $n \geq 5$, the automaton \mathcal{A}_n is strongly connected and synchronizing, and its reset threshold is at most $3n - 4$.*

Proof. We can avoid all even-indexed states on the a -cycle by word $(ab)^k$ for some k :

$$Q \xrightarrow{ab} Q \setminus \{q_4\} \xrightarrow{ab} Q \setminus \{q_4, q_6\} \xrightarrow{ab} \dots \xrightarrow{ab} \{q_1, q_2, q_3\} \cup \{q_5, q_7, \dots\}.$$

Moreover, if n is even, then we can continue applying word ab to avoid also all odd-indexed states on the a -cycle:

$$\{q_1, q_2, q_3\} \cup \{q_5, q_7, \dots\} \xrightarrow{ab} \{q_1, q_2, q_3\} \cup \{q_7, q_9, \dots\} \xrightarrow{ab} \dots \xrightarrow{ab} \{q_1, q_2, q_3\} \xrightarrow{ab} \{q_1, q_3\}.$$

If n is odd, then we can apply one additional a letter to avoid all odd-number states and repeat the procedure of avoiding even-number states, that is:

$$\begin{aligned} \{q_1, q_2, q_3\} \cup \{q_5, q_7, \dots\} &\xrightarrow{a} \{q_1, q_2, q_3\} \cup \{q_4, q_6, \dots\} \xrightarrow{ab} \{q_1, q_2, q_3\} \cup \{q_6, q_8, \dots\} \\ &\xrightarrow{ab} \dots \xrightarrow{ab} \{q_1, q_2, q_3, q_{n-1}\} \xrightarrow{ab} \{q_1, q_2, q_3\}. \end{aligned}$$

Overall, we can compress the automaton to the set $\{q_1, q_2, q_3\}$ using a word of length at most $2(n-3)$ for an even and $2(n-3)+1$ for an odd n , since each ab application decreases the size of the image by 1 until there are only three states left. The set $\{q_1, q_2, q_3\}$ can be easily synchronized to q_1 by the word $ab^{n-3}aba$. ◀

Informally, the key property of the construction is the following. To avoid q_1 , we must avoid both q_2 and q_3 . But to do so, we must first avoid two consecutive states in the cycle q_4, \dots, q_n on a . This requires one full round on this cycle. In particular, the avoided states on the cycle will be always q_4 and q_5 at some point. Then, we need a second round to map these two gaps to q_2 and q_3 , respectively. Hence, the shortest avoiding words for q_1 have length $\sim 2n$.

► **Theorem 4.** For $n \geq 7$, the 1-avoiding threshold of \mathcal{A}_n equals $2n-3$.

Proof. We show that the length of the shortest avoiding words for state q_1 is $2n-3$.

If n is even, then $(ab)^{n-3}bba$ is an avoiding word for q_1 of length $2n-3$. We have:

$$Q \xrightarrow{(ab)^{n-3}} \{q_1, q_2, q_3\} \xrightarrow{bba} \{q_2, q_4, q_5\}.$$

If $n \geq 9$ is odd, then $aba^{n-5}bab^{n-3}a$ is the desired word. We have:

$$Q \xrightarrow{ab} Q \setminus \{q_4\} \xrightarrow{a^{n-5}} Q \setminus \{q_3, q_{n-1}\} \xrightarrow{ba} Q \setminus \{q_3, q_4, q_5\} \xrightarrow{b^{n-3}} Q \setminus \{q_n, q_2, q_3\} \xrightarrow{a} Q \setminus \{q_1, q_4, q_3\}$$

For $n=7$, a shortest avoiding word is $abaabaaabba$.

To prove the lower bound, we start with an auxiliary claim.

Claim 1: If a word v avoids state q_n , then $|v| \geq n-2$. It follows because the only state that can be avoided with one letter (a) is q_3 , and the shortest words with an action mapping q_3 to q_n have length $n-3$.

Let w be a shortest word avoiding state q_1 . Since $\delta(q_1, b) = q_1$, $\delta(q_1, aa) = q_1$, and $\delta(q_1, aba) = q_1$, it follows that w must end with bba , as otherwise it would not be a shortest such word. Let write $w = w'bba$. We have $\delta(q_n, bba) = \delta(q_{n-1}, bba) = q_1$, which implies that w' must avoid both q_n and q_{n-1} . From Claim 1, $|w'| \geq n-2$, so we can write $w = w''xyubba$, where $|u| = n-5$, $|x| = |y| = 1$, and $|w''| \geq 1$. Since $w' = w''xyu$ avoids both q_n and q_{n-1} , and these states are mapped respectively to q_n and q_{n-1} by the action of every word of length $n-5$, we know that $w''xy$ avoids both q_5 and q_4 .

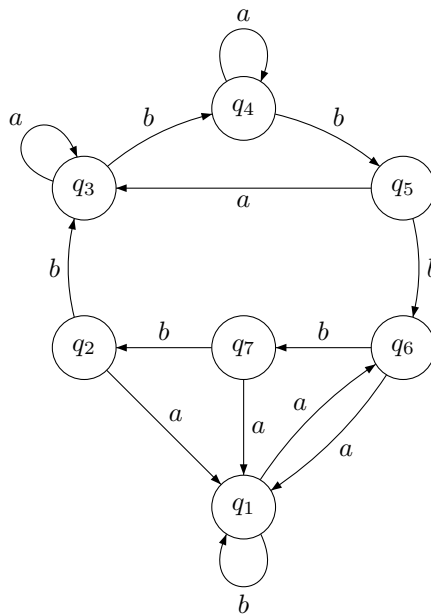
We consider three cases:

1. If $y = a$, then $w''x$ must avoid q_n (as $\delta(q_n, a) = q_4$). From Claim 1, we get that $|w''x| \geq n - 2$, hence $|w| \geq (n - 2) + 1 + (n - 5) + 3 = 2n - 3$.
2. If $y = b$ and $x = a$, then w'' must avoid q_n (as $\delta(q_n, ab) = q_5$). Similarly as in the previous case, we get that $|w| \geq 2n - 2$.
3. If $y = b$ and $x = b$, then w'' must avoid both q_2 and q_3 . Then, however, $w''a$ would avoid q_1 , which contradicts that w is a shortest such word. ◀

The proof covers the cases $n \geq 7$, whereas the lower bound for the cases $n \leq 6$ was confirmed by experiments [7].

Automata \mathcal{A}_n have another extremal property: The quadratic upper bound on the 1-avoiding threshold is derived by an iterative application of [12, Lemma 3], which for a given subset S gives a word w of length at most $n - |S| + 1$ that either avoids a fixed state $q \in Q$ or compresses S (i.e., $|\delta(S, w)| < |S|$). In the worst case, we must apply this lemma a linear number of times, hence a quadratic upper bound follows. The automata \mathcal{A}_n demonstrate that this may be the case: we may be forced to apply the lemma $\Theta(n)$ times obtaining each time the word $w = ba$, which compresses the subset. On the other hand, ba is very short compared to the linear upper bound $n - |S| + 1$. Yet, it would be possible that also this bound can be met. There also exists another series of automata (similar to the automaton from Fig. 4) showing that the upper bound $n - |S| + 1$ on the length of shortest avoiding or compressing word is tight for each $n \geq |S| \geq 1$. However, it is an open question whether both these bounds can be met simultaneously.

2.1 Exceptional examples



■ **Figure 3** An automaton with the 1-avoiding threshold equal to $2n - 2 = 12$ (for state q_1).

There are several particular examples of synchronizing automata with 1-avoiding threshold equal to $2n - 2$. For instance, for the automaton \mathcal{A}_5 it is 8. Another example is shown Fig. 3, which is a largest known automaton meeting this bound.

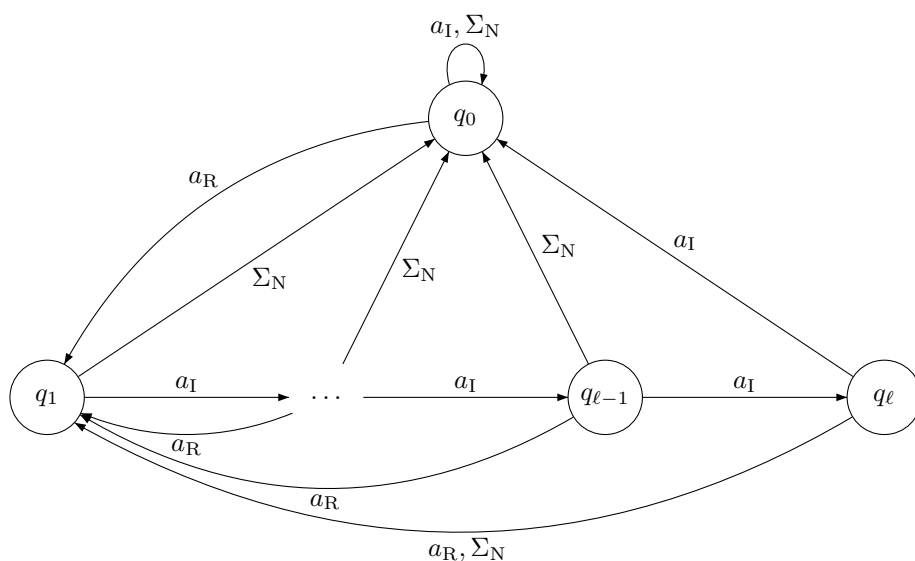
No such example seems to belong to a series keeping that 1-avoiding threshold. It is known that there are no binary synchronizing automata exceeding 1-avoiding threshold $2n - 3$ in range $n \in \{8, 9, 10, 11\}$ [7] (the case of $n = 11$ was verified later).

3 k -avoiding thresholds

3.1 General lower bound

For every fixed $k \geq 1$, we show an infinite series of (non-synchronizing) automata such that their k -avoiding threshold is in $\Theta(n^k)$. The construction is built from gadgets of two types.

3.1.1 One-track counting gadget



■ **Figure 4** The one-track counting gadget. The identity action of Σ_P is not drawn.

Let $\ell \geq 2$ be an integer, Σ_P, Σ_N be disjoint sets of letters, and $a_R, a_I \notin \Sigma_P \cup \Sigma_N$ be two other distinct letters. We define the *one-track counting gadget* $\mathcal{T}(\ell, a_R, \Sigma_P, a_I, \Sigma_N)$ (shown in Fig. 4), which is the automaton $(Q_{\mathcal{T}}, \Sigma_{\mathcal{T}}, \delta_{\mathcal{T}})$, where $P = \{q_0, q_1, \dots, q_{\ell}\}$, $\Sigma_{\mathcal{T}} = \{a_R, a_I\} \cup \Sigma_P \cup \Sigma_N$ and $\delta_{\mathcal{T}}$ is defined as follows. Letter a_R is the *reset letter* with the action mapping all the states to q_1 :

$$\delta_{\mathcal{T}}(q_i, a_R) = q_1 \text{ for } i \in \{0, 1, \dots, \ell\}.$$

Letter a_I is the *incrementing letter* with the action shifting the states q_1, \dots, q_{ℓ} :

$$\delta_{\mathcal{T}}(q_i, a_I) = q_{i+1} \text{ for } i \in \{1, \dots, \ell - 1\}; \quad \delta_{\mathcal{T}}(q_{\ell}, a_I) = q_0; \quad \delta_{\mathcal{T}}(q_0, a_I) = q_0.$$

The letters from Σ_P are called *previous letters* and they all act as the identity. The letters from Σ_N are called *next letters*; they have the same action mapping all the states to q_0 except q_{ℓ} , which is mapped to q_1 :

$$\delta_{\mathcal{T}}(q_i, a) = q_0 \text{ for } i \in \{0, 1, \dots, \ell - 1\}, a \in \Sigma_N; \quad \delta_{\mathcal{T}}(q_{\ell}, a) = q_1 \text{ for } a \in \Sigma_N.$$

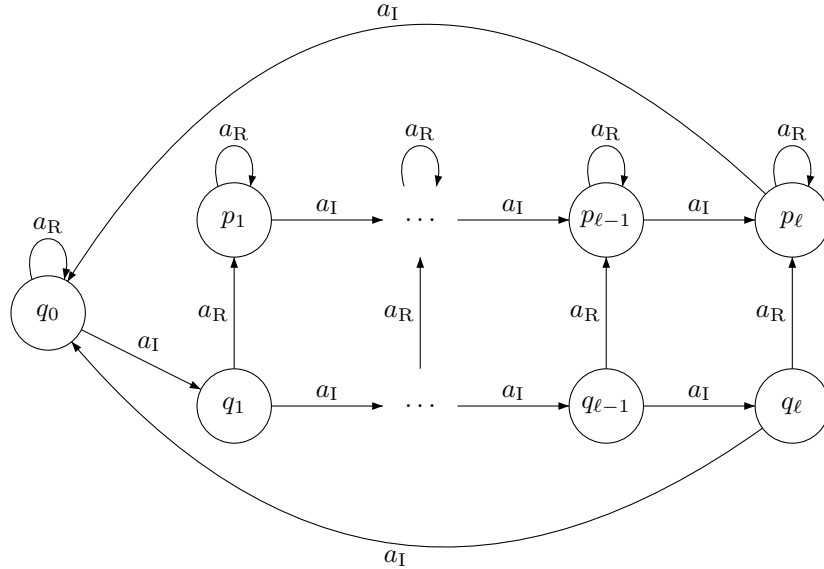
The mechanism of the gadget is that, in order to avoid q_0 , applications of next letters are restricted. We must start with a_R and then keep applying $a_I^{\ell-1}$ alternatingly with one next letter. Additionally, previous letters can be applied interleaving at any time. Formally, we have the following:

► **Lemma 5.** *Consider \mathcal{F} and a word $w \in (\{a_I\} \cup \Sigma_P \cup \Sigma_N)^*$. If the word $a_R w$ avoids q_0 , then in w , the number of occurrences of a_I is at least $\ell - 1$ times larger than the total number of occurrences of letters from Σ_N .*

Proof. In the analysis, we can ignore the occurrences of Σ_P and assume equivalently that $w \in (\{a_I\} \cup \Sigma_N)^*$.

Since w does not contain a_R and this is the only letter with the action mapping q_0 to another state, for every prefix w' of w , $a_R w'$ must also avoid q_0 . In the beginning, $\delta_{\mathcal{F}}(Q_{\mathcal{F}}, a_R) = \{q_1\}$. The only possibility to keep q_0 avoided is to apply $a_I^{\ell-1}$, which must be followed by a letter from Σ_N . We end with the singleton $\{q_1\}$ and the argument repeats, keeping the proportion between the numbers of occurrences of a_I and of letters from Σ_N . ◀

3.1.2 Two-track counting gadget



■ **Figure 5** The two-track counting gadget. The identity action of Σ_P is not drawn.

Let $\ell \geq 1$ be an integer, Σ_P be a set of letters, and a_R, a_I be two other distinct letters. We define the *two-track counting gadget* $\mathcal{D}(\ell, a_R, \Sigma_P, a_I)$ (shown in Fig. 5), which is the automaton $(Q_{\mathcal{D}}, \Sigma_P \cup \{a_R, a_I\}, \delta_{\mathcal{D}})$, where $Q_{\mathcal{D}} = \{q_0, q_1, \dots, q_\ell, p_1, \dots, p_\ell\}$, and $\delta_{\mathcal{D}}$ is defined as follows. Letter a_R is the *reset letter*, whose action maps the corresponding states q_i to p_i :

$$\delta_{\mathcal{D}}(q_i, a_R) = p_i \text{ for } i \in \{1, \dots, \ell\}; \quad \delta_{\mathcal{D}}(p_i, a_R) = p_i \text{ for } i \in \{1, \dots, \ell\}; \quad \delta_{\mathcal{D}}(q_0, a_R) = q_0.$$

Letter a_I is the *incrementing letter* acting as follows:

$$\delta_{\mathcal{D}}(q_i, a_I) = q_{i+1}, \quad \delta_{\mathcal{D}}(p_i, a_I) = p_{i+1} \text{ for } i \in \{1, \dots, \ell - 1\}; \quad \delta_{\mathcal{D}}(p_\ell, a_I) = \delta_{\mathcal{D}}(q_\ell, a_I) = q_0.$$

Finally, the letters from Σ_P act as identity.

The point of the gadget is that, in order to avoid p_ℓ , we have to apply $\ell - 1$ times letter a_I without applying letter a_R in between.

► **Lemma 6.** For \mathcal{D} , if a word $w \in \Sigma_{\mathcal{D}}^*$ avoids p_ℓ , then w contains at least ℓ occurrences of a_I without any occurrence of a_R in between.

Proof. Observe that for every word $u \in \Sigma_{\mathcal{D}}^*$, we have $\delta_{\mathcal{D}}(Q_{\mathcal{D}}, ua_R) = \{q_0, p_1, \dots, p_\ell\}$. Thus, if p_ℓ is avoided by w , then w must contain a subword (factor) $w' \in \Sigma_{\mathcal{D}} \setminus \{a_R\}$ such that $p_\ell \notin \delta_{\mathcal{D}}(\{q_0, p_1, \dots, p_\ell\}, v)$. Ignoring the letters from Σ_P , the only such words are a_i^i for $i \geq \ell$. ◀

3.1.3 The construction

We build the construction as a union of gadgets. For each $k \geq 1$ and $\ell \geq 2$, we build the automaton $\mathcal{K}(k, \ell) = (Q_{\mathcal{K}(k, \ell)}, \Sigma_{\mathcal{K}(k, \ell)}, \delta_{\mathcal{K}(k, \ell)})$. Let $\Sigma_{\mathcal{K}(k, \ell)} = \{a_R, a_1, \dots, a_k\}$ be the input alphabet. The automaton is the disjoint union of the following gadgets:

$$\begin{aligned} \mathcal{D} &= \mathcal{D}(\ell, a_R, \{a_k, \dots, a_2\}, a_1); \\ \mathcal{F}_i &= \mathcal{F}(\ell, a_R, \{a_k, \dots, a_{i+1}\}, a_i, \{a_{i-1}, \dots, a_1\}) \text{ for all } i \in \{2, \dots, k\}. \end{aligned}$$

As it contains states from several gadgets, when denoting a state, we specify the owning gadget in the superscript. Finally, we define the subset to be avoided:

$$S_k = \{p_\ell^{\mathcal{D}}\} \cup \bigcup_{i \in \{1, \dots, k-1\}} \{q_0^{\mathcal{F}_i}\}.$$

Observe that the number of states n of $\mathcal{K}(k, \ell)$ equals $(k-1)(\ell+1) + (1+2\ell) = k\ell + k + \ell = \ell(k+1) + k$.

Note that for every k , we have $Q_{\mathcal{K}(k, \ell)} \subsetneq Q_{\mathcal{K}(k+1, \ell)}$, $\Sigma_{\mathcal{K}(k, \ell)} \subsetneq \Sigma_{\mathcal{K}(k+1, \ell)}$, $\delta_{\mathcal{K}(k, \ell)} \subsetneq \delta_{\mathcal{K}(k+1, \ell)}$, and also $S_k \subsetneq S_{k+1}$. Hence, every letter of $\mathcal{K}(k, \ell)$ acts the same on the same common states in every $\mathcal{K}(k+i)$ for $i \geq 0$.

For every word $u \in \Sigma^*$, further applying a_R yields the same fixed image, that is:

$$\delta_{\mathcal{K}(k, \ell)}(Q_{\mathcal{K}(k, \ell)}, ua_R) = \delta_{\mathcal{K}(k, \ell)}(Q_{\mathcal{K}(k, \ell)}, a_R) = \{q_0^{\mathcal{D}}\} \cup \bigcup_{j \in \{1, \dots, \ell\}} \{p_j^{\mathcal{D}}\} \cup \bigcup_{i \in \{1, \dots, k\}} \{q_1^{\mathcal{F}_i}\}. \quad (1)$$

► **Lemma 7.** For $\mathcal{K}(k, \ell)$, the subset S_k is avoidable and the length of the shortest avoiding words for S_k equals $1 + \ell^k$ (and ℓ if $k = 1$).

Proof. For $k = 1$, the shortest avoiding word for S_1 is a_1^ℓ . For the remaining part, assume that $k \geq 2$.

Let w_k be a shortest avoiding word for S_k in $\mathcal{K}(k, \ell)$. First, we observe that w_k must contain a_R , since otherwise the states $q_0^{\mathcal{F}_i}$ could not be avoided. From (1), we know that w_k may contain only one occurrence of a_R and it must appear at the beginning; otherwise, there would exist a shorter avoiding word.

To show that S_k is avoidable with a word of length $1 + \ell^k$, we use induction on k . We show that there is an avoiding word w_k from $a_R \{a_1, \dots, a_k\}^*$ of the required length. For $k = 1$, there is only the gadget \mathcal{D} , and the word $w_1 = a_R a_1^\ell$ does the job. Assuming the statement for k , we show that it holds for $k+1$. Then w_k acts the same in $\mathcal{K}(k+1, \ell)$ as in $\mathcal{K}(k, \ell)$ on the common states. Let w_{k+1} be the word obtained from w_k by inserting $a_{k+1}^{\ell-1}$ before each occurrence of every letter from $\{a_1, \dots, a_k\}$. Since a_{k+1} works as the identity in the gadgets $\mathcal{D}, \mathcal{F}_1, \dots, \mathcal{F}_k$, S_k is avoided. Also, we can see that $\delta_{\mathcal{K}(k+1, \ell)}(\{q_0^{\mathcal{F}_{k+1}}, \dots, q_\ell^{\mathcal{F}_{k+1}}\}, w_{k+1}) = \{q_1^{\mathcal{F}_{k+1}}\}$. Thus w_{k+1} avoids S_{k+1} and has length $1 + \ell^k \cdot \ell = 1 + \ell^{k+1}$.

To show that there are no shorter avoiding words, we also use induction on k . We show that every avoiding word for S_k contains at least ℓ^k occurrences of letters from $\{a_1, \dots, a_k\}$. For $k = 1$, by Lemma 6, w has at least ℓ occurrences of a_1 . Assuming the statement for k ,

we show that it holds for $k + 1$. Let $w_{k+1} = a_R w'_{k+1}$ be a shortest avoiding word for S_{k+1} . Since $S_k \subset S_{k+1}$, w_{k+1} also avoids S_k . Let w'_k be the word obtained from w'_{k+1} by removing every occurrence of a_{k+1} . Then, $a_R w'_k$ is over the alphabet of $\mathcal{K}(k, \ell)$ and is an avoiding word for S_k in $\mathcal{K}(k, \ell)$. Thus, by the inductive assumption, it has at least ℓ^k occurrences of letters from $\{a_1, \dots, a_k\}$. Then w'_{k+1} contains them as well. By Lemma 4 for the last gadget \mathcal{T}_{k+1} , where the set of the next letters is $\{a_1, \dots, a_k\}$, we get that w'_{k+1} also must contain at least $(\ell - 1) \cdot \ell^k$ occurrences of a_{k+1} . Altogether, w'_{k+1} contains at least ℓ^{k+1} occurrences of letters from $\{a_1, \dots, a_{k+1}\}$. ◀

3.1.4 Strong connectivity

Each particular gadget is already strongly connected. We can make the whole construction strongly connected by redefining the special action of a_R so that its transitions work cyclically on the gadgets. Let

$$\begin{aligned} \delta_{\mathcal{K}(k, \ell)}(q_0^{\mathcal{D}}, a_R) &= q_1^{\mathcal{T}_1}; \\ \delta_{\mathcal{K}(k, \ell)}(q_j^{\mathcal{T}_i}, a_R) &= q_1^{\mathcal{T}_{i+1}} \text{ for } i \in \{1, \dots, k-1\}, j \in \{0, \dots, \ell\}; \\ \delta_{\mathcal{K}(k, \ell)}(q_j^{\mathcal{T}_k}, a_R) &= q_0^{\mathcal{D}} \text{ for } j \in \{0, \dots, \ell\}; \end{aligned}$$

and the action is left unchanged for the other states of \mathcal{D} : $q_1^{\mathcal{D}}, \dots, q_\ell^{\mathcal{D}}, p_1^{\mathcal{D}}, \dots, p_\ell^{\mathcal{D}}$.

Since (1) still holds for the modified construction, Lemma 7 works as well. We conclude the construction with the following:

► **Theorem 8.** *For every $k \geq 2$, there exists an infinite series of strongly connected automata such that its k -avoiding threshold is at least $1 + \left(\frac{n-k}{k+1}\right)^k$. For a fixed k , its k -avoiding threshold is in $\Theta(n^k)$.*

Proof. For an integer $k \geq 2$, we build the automata $\mathcal{K}(k, \ell)$ for all $\ell \geq 2$. Each $\mathcal{K}(k, \ell)$ has $n = \ell(k+1) + k$ states and its k -avoiding threshold is at least $1 + \ell^k$, thus we get the lower bound.

Note that a_R is the shortest word of the minimal rank, so for a fixed k , the lower bound asymptotically coincides with the upper bound from Theorem 2 for $k \geq 2$, thus the k -avoiding threshold is in $\Theta(n^k)$. ◀

3.2 2-avoiding threshold reduction

The bound $\mathcal{O}(n^k)$ is asymptotically tight for $k \geq 3$, but the cases of $k = 1$ and $k = 2$ remain open. This is due to the cubic upper bound on the length of the shortest minimal-rank words. However, we can reduce the problem for $k = 2$ to the case of $k = 1$. If 1-avoiding threshold is bounded by $f(n)$, then 2-avoiding threshold is at most $\mathcal{O}(n^2 + n \cdot f(n))$. Thus, if 1-avoiding threshold is at most linear, then 2-avoiding threshold is at most quadratic, which would be a tight bound.

A similar result, but restricted to strongly connected synchronizing automata, was shown in [12, Lemma 13]. Here, we show it in the general case, which in essence combines both techniques behind Theorem 8 and Theorem 4.

► **Theorem 9.** *For every n -state automaton, if the 1-avoiding threshold is at most $f(n)$ for some function f , then the 2-avoiding threshold is at most $\mathcal{O}(n^2 + n \cdot f(n))$.*

Proof. A *strongly connected bottom component* is a minimal non-empty set of states $X \subseteq Q$ such that for every word w we have $\delta(X, w) \subseteq X$. A synchronizing automaton has exactly one strongly connected bottom component, whereas a non-synchronizing automaton can have many of them. Let z be a shortest word such that all states in $\delta(Q, z)$ are in the strongly connected bottom components of the automaton. It is well-known that the length of z is in $\mathcal{O}(n^2)$ [9].

Let $\{q_1, q_2\}$ be a subset to avoid. States q_1 and q_2 are either in the same strongly connected component or in separate ones. In any case, we can ignore every other strongly connected component, since their states cannot be mapped to q_1 or q_2 . Let C_1 and C_2 be these components, respectively of q_1 and q_2 . From now, consider the automaton containing only C_1 and C_2 .

Let u_1 and u_2 be avoiding words respectively for q_1 and q_2 , both of length at most $f(n)$.

We build iteratively some words w_1, w_2, \dots, w_{n-1} . Each w_i will be of length $\mathcal{O}(i(n+f(n)))$. Let $w_0 = \varepsilon$.

Assume that we have built w_i , and let $X = \delta(Q, w_i)$. Now, we use a linear algebraic argument to infer that one of three possibilities hold:

- (1) there exists a word v of length at most $n - 1$ such that $q_1 \notin \delta(X, vu_2)$, or
- (2) there exists a word v of length at most $n - 1$ such that there are at least two distinct states $r_1, r_2 \in X$ such that $\delta(r_1, vu_2) = \delta(r_2, vu_2) = q_1$, or
- (3) there does not exist any word v (of any length) satisfying (1) or (2).

We omit to repeat the argument here since it follows in the same way as in the proof of [12, Lemma 13] and requires introducing many linear algebraic definitions. In [12, Lemma 13], however, (3) cannot happen due to the assumption that the automaton is synchronizing and strongly connected, so such a word v always exist.

If (1) holds, then $w_i vu_2$ is an avoiding word for $\{q_1, q_2\}$ and it has a desired length. If (2) holds, then let $w_{i+1} = w_i vu_2$, which is longer than w_i by at most $n - 1 + f(n)$. If (3) holds, then it means that w_i in the automaton restricted to C_1 has the minimal rank. Otherwise, we could map two distinct states from $X \cap C_1$ to the same state, and then map it to q_1 . In this case, we also stop with w_i .

If we have stopped with (3), then we repeat the construction symmetrically for q_2 . If we also do not find an avoiding word, then we obtain a word w'_i that has the minimal rank in the automaton restricted to C_2 . Altogether, $w_i w'_i$ has the minimal rank the automaton with both C_1 and C_2 .

Then, it remains to use the upper bound from Theorem 2. Since the length of $w_i w'_i$ is an upper bound on m , we get the upper bound $\mathcal{O}(n^2 + n \cdot f(n))$.

Finally, we have to add z at the beginning to construct an avoiding word in the original automaton, and the length of z is also at most $\mathcal{O}(n^2)$. ◀

4 Bounding reset threshold with avoiding words

If 1-avoiding threshold is subquadratic, it would yield an upper bound on reset threshold equal to $7/48n^3 + o(n^3)$. However, the application of 1-avoiding threshold to bound reset threshold can be generalized to the usage of k -avoiding thresholds, if they are small enough. It turns out that, in a synchronizing automaton, a subquadratic value of k -avoiding thresholds already for small values of k is enough to imply a subcubic upper bound on the reset threshold. In the following calculations, we disregard particular constants and focus only on asymptotic bounds.

46:12 Lower Bounds on Avoiding Thresholds

► **Lemma 10.** *Let (Q, Σ, δ) be an n -state synchronizing automaton and w be a word of rank $r \geq 2$. There exist at least one state $q \in \delta(Q, w)$ such that $|\delta^{-1}(\{q\}, w)| \leq \lfloor n/r \rfloor$.*

Proof. For each q , the states in $\delta^{-1}(\{q\}, w)$ are pairwise disjoint and they cover the whole Q . As we have r states in R , there exist a state $q \in R$ such that the cardinality of $\delta^{-1}(\{q\}, w)$ is at most n/r . ◀

► **Lemma 11.** *Let (Q, Σ, δ) be an n -state synchronizing automaton and w be a word of rank $r \geq 2$. There is a word of rank at most $r - 1$ and of length at most $|w| + d$, where d is the $\lfloor n/r \rfloor$ -avoiding threshold.*

Proof. We take the state q from Lemma 10. Since $r \geq 2$, so $\lfloor n/r \rfloor < n$, thus every subset of that size is avoidable. We first avoid the subset $\delta^{-1}(\{q\}, w)$ by a word u of length at most d , and then apply w . We have $\delta(Q, uw) \subseteq \delta(Q, w)$ and $q \in \delta(Q, w)$, but also $q \notin \delta(Q, uw)$, thus we obtain that $\delta(Q, uw) \subsetneq \delta(Q, w)$. ◀

Taking the usual notation, $\omega(1)$ is the set of $\mathbb{R} \rightarrow \mathbb{R}$ functions growing faster than a constant.

► **Theorem 12.** *If there exists a function $f \in \omega(1)$ such that for every n -state synchronizing automaton and every $k \leq f(n)$, the k -avoiding threshold is at most $\mathcal{O}(n^2)/f(n)$, then the reset threshold is in $o(n^3)$.*

Proof. We build a reset word in two phases. First, we start with the empty word and iteratively apply Lemma 11 until the built word reaches a rank of at most $n/f(n)$. Thus, there are at most $n - n/f(n)$ applications of the lemma. The rank is at least $n/f(n)$ every time, so we need the k -avoiding threshold for $k \leq n/(n/f(n)) = f(n)$. Hence, d from the lemma is bounded above by $\mathcal{O}(n^2)/f(n)$ from the assumption of the theorem. It follows that the resulted word of rank $n/f(n)$ has length at most $(n - n/f(n)) \cdot \mathcal{O}(n^2)/f(n) = \mathcal{O}(n^3)/f(n)$.

In the second phase, we use the usual pair compression [8]. We need to compress a pair at most $n/f(n) - 1$ times, each time appending a word of length smaller than n^2 . Thus, the word from this phase also has length at most $\mathcal{O}(n^3)/f(n)$.

Altogether, our reset word has length at most $n^3/f(n)$, which is in $o(n^3)$. ◀

5 Conclusions, discussion, and open problems

In the general case of an automaton and its k -avoiding threshold, we have a complete asymptotic solution for $k \geq 3$. The cases of $k = 2$ and $k = 1$ are open, yet a conjectured solution to the latter would solve also the former.

As for now, determining the maximum 1-avoiding threshold is the core problem, in particular in the class of synchronizing automata, but the general case of an automaton does not seem different for $k = 1$. Here, we considered a precise bound and have shown that $2n - 3$ is attainable for every $n \geq 2$. Yet, a few isolated examples reach $2n - 2$. We have the following conjecture:

A *trivial extension* of an automaton (Q, Σ, δ) is any automaton $(Q, \Sigma \cup \Sigma', \delta \cup \delta')$ such that each letter from Σ' act either as an identity or as a letter from Σ .

► **Conjecture 13.** *For an n -state automaton, the 1-avoiding threshold is at most $2n - 3$, except for a finite number of cases and their trivial extensions, where it is equal to $2n - 2$.*

A weak version (with any linear bound) of Conjecture 13 implies the following:

► **Conjecture 14.** *For an n -state automaton, a tight upper bound on 2-avoiding threshold is $\mathcal{O}(n^2)$.*

Conjecture 13 was verified for small cases [7], in particular up to 11 states for binary synchronizing automata. We have also experimented with the case of $k = 2$. The maximum 2-avoiding threshold of an n -state binary synchronizing automaton for $n = 3, \dots, 10$ equals respectively 6, 8, 12, 17, 19, 23, 25, and 28. However, for this problem, the range is much too small to reveal the tendency.

The case of synchronizing automata is much harder, and the problem remains open for all k . We have the trivial upper bound $\mathcal{O}(n^3)$, thus the situation is surely different than in the general case already for $k \geq 4$ and most likely already for $k \geq 2$.

We have made an effort to find a lower bound on the maximum possible k -avoiding threshold also for $k \geq 2$. We have not found any series of automata that would exceed the upper bound $\mathcal{O}(kn)$ (when k is a variable). This bound is easily met, for example by the well-known Rystsov series [10] with a sink (zero) state and reset threshold $n(n-1)/2$. We should consider the following:

► **Conjecture 15.** *For an n -state synchronizing automaton, the k -avoiding threshold is at most $\mathcal{O}(kn)$ (when k is a variable).*

It turns out that this is a weaker (by several means) version of the disproved Don's conjecture [3, Conjecture 18], which states that if a subset T is reachable, then it is reachable with a word of length at most $n(n - |T|)$. We know that Conjecture 15 does not hold for non-synchronizing automata, and even for synchronizing automata with the original non-asymptotic bound (by e.g., Theorem 4 for $k = 1$). On the other hand, if the reset threshold is at most quadratic, then the conjecture trivially holds for $k \in \Theta(n)$.

Finally, we have noted that a subquadratic upper bound on k -avoiding threshold of a synchronizing automaton for small, but non-constant, values of k , is enough to imply a subcubic upper bound on reset threshold. This should motivate further efforts on bounding avoiding thresholds. Since the current upper bound on 1-avoiding threshold is quadratic, this requires both decreasing it and generalizing to k -avoiding thresholds for small k . We know that a subquadratic upper bound for $k = 2$ is not possible in the class of non-synchronizing automata. Yet, our conjectures are much stronger than the required upper bound to give further improvements.

References



- 1 M. V. Berlinkov, R. Ferens, and M. Szykuła. Preimage problems for deterministic finite automata. *Journal of Computer and System Sciences*, 115:214–234, 2021.
- 2 J. Černý. Poznámka k homogénnym experimentom s konečnými automatami. *Matematicko-fyzikálny Časopis Slovenskej Akadémie Vied*, 14(3):208–216, 1964. In Slovak.
- 3 H. Don. The Černý Conjecture and 1-Contracting Automata. *Electronic Journal of Combinatorics*, 23(3):P3.12, 2016.
- 4 F. Gonze and R. M. Jungers. Hardly Reachable Subsets and Completely Reachable Automata with 1-Deficient Words. *Journal of Automata, Languages and Combinatorics*, 24(2–4):321–342, 2019.
- 5 F. Gonze, R. M. Jungers, and A. N. Trahtman. A Note on a Recent Attempt to Improve the Pin-Frankl Bound. *Discrete Mathematics and Theoretical Computer Science*, 17(1):307–308, 2015.
- 6 J. Kari. Synchronizing finite automata on Eulerian digraphs. *Theoretical Computer Science*, 295(1-3):223–232, 2003.

- 7 A. Kisielewicz, J. Kowalski, and M. Szykuła. Experiments with Synchronizing Automata. In *Implementation and Application of Automata*, volume 9705 of *LNCS*, pages 176–188. Springer, 2016.
- 8 J.-E. Pin. On two combinatorial problems arising from automata theory. In *Proceedings of the International Colloquium on Graph Theory and Combinatorics*, volume 75 of *North-Holland Mathematics Studies*, pages 535–548, 1983.
- 9 I. K. Rystsov. Polynomial complete problems in automata theory. *Information Processing Letters*, 16(3):147–151, 1983.
- 10 I. K. Rystsov. Quasioptimal Bound for the Length of Reset Words for Regular Automata. *Acta Cybernetica*, 12(2):145–152, 1995.
- 11 Y. Shitov. An Improvement to a Recent Upper Bound for Synchronizing Words of Finite Automata. *Journal of Automata, Languages and Combinatorics*, 24(2–4):367–373, 2019.
- 12 M. Szykuła. Improving the Upper Bound on the Length of the Shortest Reset Word. In *STACS 2018, LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.
- 13 A. N. Trahtman. The Černý conjecture for aperiodic automata. *Discrete Mathematics and Theoretical Computer Science*, 9(2):3–10, 2007.
- 14 A. N. Trahtman. Modifying the upper bound on the length of minimal synchronizing word. In *Fundamentals of Computation Theory*, volume 6914 of *LNCS*, pages 173–180. Springer, 2011.
- 15 M. V. Volkov. Synchronizing automata and the Černý conjecture. In *Language and Automata Theory and Applications*, volume 5196 of *LNCS*, pages 11–27. Springer, 2008.
- 16 M. V. Volkov. Synchronizing automata preserving a chain of partial orders. *Theoretical Computer Science*, 410(37):3513–3519, 2009.
- 17 M. V. Volkov, editor. *Special Issue: Essays on the Černý Conjecture*, volume 24 (2–4) of *Journal of Automata, Languages and Combinatorics*, 2019.
- 18 V. Vorel. Subset synchronization and careful synchronization of binary finite automata. *Int. J. Found. Comput. Sci.*, 27(5):557–578, 2016.

HyperLTL Satisfiability Is Σ_1^1 -Complete, HyperCTL* Satisfiability Is Σ_1^2 -Complete

Marie Fortin  



University of Liverpool, UK

Louwe B. Kuijer  

University of Liverpool, UK

Patrick Totzke  

University of Liverpool, UK

Martin Zimmermann  

University of Liverpool, UK

Abstract

Temporal logics for the specification of information-flow properties are able to express relations between multiple executions of a system. The two most important such logics are HyperLTL and HyperCTL*, which generalise LTL and CTL* by trace quantification. It is known that this expressiveness comes at a price, i.e. satisfiability is undecidable for both logics.

In this paper we settle the exact complexity of these problems, showing that both are in fact highly undecidable: we prove that HyperLTL satisfiability is Σ_1^1 -complete and HyperCTL* satisfiability is Σ_1^2 -complete. These are significant increases over the previously known lower bounds and the first upper bounds. To prove Σ_1^2 -membership for HyperCTL*, we prove that every satisfiable HyperCTL* sentence has a model that is equinumerous to the continuum, the first upper bound of this kind. We prove this bound to be tight. Finally, we show that the membership problem for every level of the HyperLTL quantifier alternation hierarchy is Π_1^1 -complete.

2012 ACM Subject Classification Theory of computation \rightarrow Logic and verification; Theory of computation \rightarrow Formal languages and automata theory

Keywords and phrases HyperLTL, HyperCTL*, Satisfiability, Analytical Hierarchy

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.47

Related Version *Extended Version*: <https://arxiv.org/abs/2105.04176> [27]

Funding Partially funded by EPSRC grants EP/S032207/1 and EP/V025848/1.

Acknowledgements We thank Karoliina Lehtinen and Wolfgang Thomas for fruitful discussions.

1 Introduction

Most classical temporal logics like LTL and CTL* refer to a single execution trace at a time while information-flow properties, which are crucial for security-critical systems, require reasoning about multiple executions of a system. Clarkson and Schneider [13] coined the term *hyperproperties* for such properties which, structurally, are sets of *sets* of traces. Just like ordinary trace and branching-time properties, hyperproperties can be specified using temporal logics, e.g. HyperLTL and HyperCTL* [12], expressive, but intuitive specification languages that are able to express typical information-flow properties such as noninterference, noninference, declassification, and input determinism. Due to their practical relevance and theoretical elegance, hyperproperties and their specification languages have received considerable attention during the last decade [1, 2, 5, 6, 7, 10, 12, 13, 14, 16, 26, 28, 31, 32, 39].



© Marie Fortin, Louwe B. Kuijer, Patrick Totzke, and Martin Zimmermann;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 47; pp. 47:1–47:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

HyperLTL is obtained by extending LTL [34], the most influential specification language for linear-time properties, by trace quantifiers to refer to multiple executions of a system. For example, the HyperLTL formula

$$\forall\pi. \forall\pi'. \mathbf{G}(i_\pi \leftrightarrow i_{\pi'}) \rightarrow \mathbf{G}(o_\pi \leftrightarrow o_{\pi'})$$

expresses input determinism, i.e. every pair of traces that always has the same input (represented by the proposition i) also always has the same output (represented by the proposition o). Similarly, HyperCTL* is the extension of the branching-time logic CTL* [17] by path quantifiers. HyperLTL only allows formulas in prenex normal form while HyperCTL* allows arbitrary quantification, in particular under the scope of temporal operators. Consequently, HyperLTL formulas are evaluated over sets of traces while HyperCTL* formulas are evaluated over transition systems, which yield the underlying branching structure of the traces.

All basic verification problems, e.g. model checking [18, 25], runtime monitoring [3, 8, 11, 24], and synthesis [9, 21, 22], have been studied. Most importantly, HyperCTL* model checking over finite transition systems is decidable and TOWER-complete for a fixed transition system [25, 33]. However, for a small number of alternations, efficient algorithms have been developed and were applied to a wide range of problems, e.g. an information-flow analysis of an I2C bus master [25], the symmetric access to a shared resource in a mutual exclusion protocol [25], and to detect the use of a defeat device to cheat in emission testing [4].

But surprisingly, the exact complexity of the satisfiability problems for HyperLTL and HyperCTL* is still open. Finkbeiner and Hahn proved that HyperLTL satisfiability is undecidable [19], a result which already holds when only considering finite sets of ultimately periodic traces and $\forall\exists$ -formulas. In fact, Finkbeiner et al. showed that HyperLTL satisfiability restricted to finite sets of ultimately periodic traces is Σ_1^0 -complete [20] (i.e. complete for the set of recursively enumerable problems). Furthermore, Hahn and Finkbeiner proved that the $\exists^*\forall^*$ -fragment has decidable satisfiability [19] while Mascle and Zimmermann studied the HyperLTL satisfiability problem restricted to bounded sets of traces [33]. The latter work implies that HyperLTL satisfiability restricted to finite sets of traces (even non ultimately periodic ones) is also Σ_1^0 -complete. Finally, Finkbeiner et al. developed tools and heuristics [20, 23].

As every HyperLTL formula can be turned into an equisatisfiable HyperCTL* formula, HyperCTL* satisfiability is also undecidable. Moreover, Rabe has shown that it is even Σ_1^1 -hard [35], i.e. it is not even arithmetical. However, both for HyperLTL and for HyperCTL* satisfiability, only lower bounds, but no upper bounds, are known.

Our Contributions. In this paper, we settle the complexity of the satisfiability problems for HyperLTL and HyperCTL* by determining exactly how undecidable they are. That is, we provide matching lower and upper bounds in terms of the analytical hierarchy and beyond, where decision problems (encoded as subsets of \mathbb{N}) are classified based on their definability by formulas of higher-order arithmetic, namely by the type of objects one can quantify over and by the number of alternations of such quantifiers. We refer to Roger's textbook [36] for fully formal definitions. For our purposes, it suffices to recall the following classes. Σ_1^0 contains the sets of natural numbers of the form

$$\{x \in \mathbb{N} \mid \exists x_0 \cdots \exists x_k. \psi(x, x_0, \dots, x_k)\}$$

where quantifiers range over natural numbers and ψ is a quantifier-free arithmetic formula. The notation Σ_1^0 signifies that there is a single block of existential quantifiers (the subscript 1) ranging over natural numbers (type 0 objects, explaining the superscript 0). Analogously, Σ_1^1

is induced by arithmetic formulas with existential quantification of type 1 objects (functions mapping natural numbers to natural numbers) and arbitrary (universal and existential) quantification of type 0 objects. Finally, Σ_1^2 is induced by arithmetic formulas with existential quantification of type 2 objects (functions mapping type 1 objects to natural numbers) and arbitrary quantification of type 0 and type 1 objects. So, Σ_1^0 is part of the first level of the arithmetic hierarchy, Σ_1^1 is part of the first level of the analytical hierarchy, while Σ_1^2 is not even analytical.

In terms of this classification, we prove that HyperLTL satisfiability is Σ_1^1 -complete while HyperCTL* satisfiability is Σ_1^2 -complete, thereby settling the complexity of both problems and showing that they are highly undecidable. In both cases, this is a significant increase of the lower bound and the first upper bound.

First, let us consider HyperLTL satisfiability. The Σ_1^1 lower bound is a reduction from the recurrent tiling problem, a standard Σ_1^1 -complete problem asking whether $\mathbb{N} \times \mathbb{N}$ can be tiled by a given finite set of tiles. So, let us consider the upper bound: Σ_1^1 allows to quantify over type 1 objects: functions from natural numbers to natural numbers, or, equivalently, over sets of natural numbers, i.e. countable objects. On the other hand, HyperLTL formulas are evaluated over sets of infinite traces, i.e. uncountable objects. Thus, to show that quantification over type 1 objects is sufficient, we need to apply a result of Finkbeiner and Zimmermann proving that every satisfiable HyperLTL formula has a countable model [26]. Then, we can prove Σ_1^1 -membership by expressing the existence of a model and the existence of appropriate Skolem functions for the trace quantifiers by type 1 quantification. We also prove that the satisfiability problem remains Σ_1^1 -complete when restricted to ultimately periodic traces, or, equivalently, when restricted to finite traces.

Then, we turn our attention to HyperCTL* satisfiability. Recall that HyperCTL* formulas are evaluated over (possibly infinite) transition systems, which can be much larger than type 2 objects whose cardinality is bounded by \mathfrak{c} , the cardinality of the continuum. Hence, to obtain our upper bound on the complexity we need, just like in the case of HyperLTL, an upper bound on the size of minimal models of satisfiable HyperCTL* formulas. To this end, we generalise the proof of Finkbeiner and Zimmermann to HyperCTL*, showing that every satisfiable HyperCTL* formula has a model of size \mathfrak{c} . We also exhibit a satisfiable HyperCTL* formula $\varphi_{\mathfrak{c}}$ whose models all have at least cardinality \mathfrak{c} , as they have to encode all subsets of \mathbb{N} by disjoint paths. Thus, our upper bound \mathfrak{c} is tight.

With this upper bound on the cardinality of models, we are able to prove Σ_1^2 -membership of HyperCTL* satisfiability by expressing with type 2 quantification the existence of a model and the existence of a winning strategy in the induced model checking game. The matching lower bound is proven by directly encoding the arithmetic formulas inducing Σ_1^2 as instances of the HyperCTL* satisfiability problem. To this end, we use the formula $\varphi_{\mathfrak{c}}$ whose models have for each subset $A \subseteq \mathbb{N}$ a path encoding A . Now, quantification over type 0 objects (natural numbers) is simulated by quantification of a path encoding a singleton set, quantification over type 1 objects (which can be assumed to be sets of natural numbers) is simulated by quantification over the paths encoding such subsets, and existential quantification over type 2 objects (which can be assumed to be subsets of $2^{\mathbb{N}}$) is simulated by the choice of the model, i.e. a model encodes k subsets of $2^{\mathbb{N}}$ if there are k existential type 2 quantifiers. Finally, the arithmetic operations can easily be implemented in HyperLTL, and therefore also in HyperCTL*.

After settling the complexity of satisfiability, we turn our attention to the HyperLTL quantifier alternation hierarchy and its relation to satisfiability. Rabe remarks that the hierarchy is strict [35], and Mascle and Zimmermann show that every HyperLTL formula has

a polynomial-time computable equi-satisfiable formula with one quantifier alternation [33]. Here, we present a novel proof of strictness by embedding the FO[<] alternation hierarchy, which is also strict [15, 37]. We use our construction to prove that for every $n > 0$, deciding whether a given formula is equivalent to a formula with at most n quantifier alternations is Π_1^1 -complete (i.e. the co-class of Σ_1^1).

All proofs omitted due to space restrictions can be found in the full version [27].

2 Preliminaries

Fix a finite set AP of atomic propositions. A *trace* over AP is a map $t: \mathbb{N} \rightarrow 2^{\text{AP}}$, denoted by $t(0)t(1)t(2)\dots$. It is *ultimately periodic*, if $t = x \cdot y^\omega$ for some $x, y \in (2^{\text{AP}})^+$, i.e. there are $s, p > 0$ with $t(n) = t(n+p)$ for all $n \geq s$. The set of all traces over AP is $(2^{\text{AP}})^\omega$.

A transition system $\mathcal{T} = (V, E, v_I, \lambda)$ consists of a set V of vertices, a set $E \subseteq V \times V$ of (directed) edges, an initial vertex $v_I \in V$, and a labelling $\lambda: V \rightarrow 2^{\text{AP}}$ of the vertices by sets of atomic propositions. A path ρ through \mathcal{T} is an infinite sequence $\rho(0)\rho(1)\rho(2)\dots$ of vertices with $(\rho(n), \rho(n+1)) \in E$ for every $n \geq 0$.

HyperLTL. The formulas of HyperLTL are given by the grammar

$$\varphi ::= \exists \pi. \varphi \mid \forall \pi. \varphi \mid \psi \qquad \psi ::= a_\pi \mid \neg \psi \mid \psi \vee \psi \mid \mathbf{X} \psi \mid \psi \mathbf{U} \psi$$

where a ranges over atomic propositions in AP and where π ranges over a fixed countable set \mathcal{V} of (*trace*) *variables*. Conjunction, implication, and equivalence are defined as usual, and the temporal operators eventually **F** and always **G** are derived as $\mathbf{F} \psi = \neg \psi \mathbf{U} \psi$ and $\mathbf{G} \psi = \neg \mathbf{F} \neg \psi$. A *sentence* is a formula without free variables.

The semantics of HyperLTL is defined with respect to a *trace assignment*, a partial mapping $\Pi: \mathcal{V} \rightarrow (2^{\text{AP}})^\omega$. The assignment with empty domain is denoted by Π_\emptyset . Given a trace assignment Π , a variable π , and a trace t we denote by $\Pi[\pi \rightarrow t]$ the assignment that coincides with Π everywhere but at π , which is mapped to t . Furthermore, $\Pi[j, \infty)$ denotes the trace assignment mapping every π in Π 's domain to $\Pi(\pi)(j)\Pi(\pi)(j+1)\Pi(\pi)(j+2)\dots$, its suffix from position j onwards.

For sets T of traces and trace assignments Π we define

- $(T, \Pi) \models a_\pi$ if $a \in \Pi(\pi)(0)$,
- $(T, \Pi) \models \neg \psi$ if $(T, \Pi) \not\models \psi$,
- $(T, \Pi) \models \psi_1 \vee \psi_2$ if $(T, \Pi) \models \psi_1$ or $(T, \Pi) \models \psi_2$,
- $(T, \Pi) \models \mathbf{X} \psi$ if $(T, \Pi[1, \infty)) \models \psi$,
- $(T, \Pi) \models \psi_1 \mathbf{U} \psi_2$ if there is a $j \geq 0$ such that $(T, \Pi[j, \infty)) \models \psi_2$ and for all $0 \leq j' < j$: $(T, \Pi[j', \infty)) \models \psi_1$,
- $(T, \Pi) \models \exists \pi. \varphi$ if there exists a trace $t \in T$ such that $(T, \Pi[\pi \rightarrow t]) \models \varphi$, and
- $(T, \Pi) \models \forall \pi. \varphi$ if for all traces $t \in T$: $(T, \Pi[\pi \rightarrow t]) \models \varphi$.

We say that T *satisfies* a sentence φ if $(T, \Pi_\emptyset) \models \varphi$. In this case, we write $T \models \varphi$ and say that T is a *model* of φ . Although HyperLTL sentences are required to be in prenex normal form, they are closed under Boolean combinations, which can easily be seen by transforming such formulas into prenex normal form. Two HyperLTL sentences φ and φ' are equivalent if $T \models \varphi$ if and only if $T \models \varphi'$ for every set T of traces.

HyperCTL*. The formulas of HyperCTL* are given by the grammar

$$\varphi ::= a_\pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U}\varphi \mid \exists\pi. \varphi \mid \forall\pi. \varphi$$

where a ranges over atomic propositions in AP and where π ranges over a fixed countable set \mathcal{V} of (*path*) *variables*, and where we require that each temporal operator appears in the scope of a path quantifier. Again, other Boolean connectives and temporal operators are derived as usual. Sentences are formulas without free variables.

Let \mathcal{T} be a transition system. The semantics of HyperCTL* is defined with respect to a *path assignment*, a partial mapping Π from variables in \mathcal{V} to paths of \mathcal{T} . The assignment with empty domain is denoted by Π_\emptyset . Given a path assignment Π , a variable π , and a path ρ we denote by $\Pi[\pi \rightarrow \rho]$ the assignment that coincides with Π everywhere but at π , which is mapped to ρ . Furthermore, $\Pi[j, \infty)$ denotes the path assignment mapping every π in Π 's domain to $\Pi(\pi)(j)\Pi(\pi)(j+1)\Pi(\pi)(j+2)\cdots$, its suffix from position j onwards.

For transition systems \mathcal{T} and path assignments Π we define

- $(\mathcal{T}, \Pi) \models a_\pi$ if $a \in \lambda(\Pi(\pi)(0))$, where λ is the labelling function of \mathcal{T} ,
- $(\mathcal{T}, \Pi) \models \neg\psi$ if $(\mathcal{T}, \Pi) \not\models \psi$,
- $(\mathcal{T}, \Pi) \models \psi_1 \vee \psi_2$ if $(\mathcal{T}, \Pi) \models \psi_1$ or $(\mathcal{T}, \Pi) \models \psi_2$,
- $(\mathcal{T}, \Pi) \models \mathbf{X}\psi$ if $(\mathcal{T}, \Pi[1, \infty)) \models \psi$,
- $(\mathcal{T}, \Pi) \models \psi_1 \mathbf{U}\psi_2$ if there exists a $j \geq 0$ such that $(\mathcal{T}, \Pi[j, \infty)) \models \psi_2$ and for all $0 \leq j' < j$: $(\mathcal{T}, \Pi[j', \infty)) \models \psi_1$,
- $(\mathcal{T}, \Pi) \models \exists\pi. \varphi$ if there exists a path ρ of \mathcal{T} , starting in $\text{rcnt}(\Pi)$, such that $(\mathcal{T}, \Pi[\pi \rightarrow \rho]) \models \varphi$, and
- $(\mathcal{T}, \Pi) \models \forall\pi. \varphi$ if for all paths ρ of \mathcal{T} starting in $\text{rcnt}(\Pi)$: $(\mathcal{T}, \Pi[\pi \rightarrow \rho]) \models \varphi$.

Here, $\text{rcnt}(\Pi)$ is the initial vertex of $\Pi(\pi)$, where π is the path variable most recently added to Π , and the initial vertex of \mathcal{T} if Π is empty.¹ We say that \mathcal{T} *satisfies* a sentence φ if $(\mathcal{T}, \Pi_\emptyset) \models \varphi$. In this case, we write $\mathcal{T} \models \varphi$ and say that \mathcal{T} is a *model* of φ .

Complexity Classes for Undecidable Problems. A type 0 object is a natural number $n \in \mathbb{N}$, a type 1 object is a function $f: \mathbb{N} \rightarrow \mathbb{N}$, and a type 2 object is a function $f: (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$. As usual, predicate logic with quantification over type 0 objects (first-order quantifiers) is called first-order logic. Second- and third-order logic are defined similarly.

We consider formulas of arithmetic, i.e. predicate logic with signature $(0, 1, +, \cdot, <)$ evaluated over the natural numbers. With a single free variable of type 0, such formulas define sets of natural numbers (see, e.g. Rogers [36] for more details):

- Σ_1^0 contains the sets of the form $\{x \in \mathbb{N} \mid \exists x_0 \cdots \exists x_k. \psi(x, x_0, \dots, x_k)\}$ where ψ is a quantifier-free arithmetic formula and the x_i are variables of type 0.
- Σ_1^1 contains the sets of the form $\{x \in \mathbb{N} \mid \exists x_0 \cdots \exists x_k. \psi(x, x_0, \dots, x_k)\}$ where ψ is an arithmetic formula with arbitrary (existential and universal) quantification over type 0 objects and the x_i are variables of type 1.
- Σ_1^2 contains the sets of the form $\{x \in \mathbb{N} \mid \exists x_0 \cdots \exists x_k. \psi(x, x_0, \dots, x_k)\}$ where ψ is an arithmetic formula with arbitrary (existential and universal) quantification over type 0 and type 1 objects and the x_i are variables of type 2.

Note that there is a bijection between functions of the form $f: \mathbb{N} \rightarrow \mathbb{N}$ and subsets of \mathbb{N} , which is implementable in arithmetic. Similarly, there is a bijection between functions of the

¹ For the sake of simplicity, we refrain from formalising this notion properly, which would require to keep track of the order in which variables are added to Π 's domain.

form $f: (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ and subsets of $2^{\mathbb{N}}$, which is again implementable in arithmetic. Thus, whenever convenient, we use quantification over sets of natural numbers and over sets of sets of natural numbers, instead of quantification over type 1 and type 2 objects; in particular when proving lower bounds. We then include \in in the signature.

3 HyperLTL satisfiability is Σ_1^1 -complete

In this section we settle the complexity of the satisfiability problem for HyperLTL: given a HyperLTL sentence, determine whether it has a model.

► **Theorem 1.** *HyperLTL satisfiability is Σ_1^1 -complete.*

We should contrast this result with [20, Theorem 1], which shows that HyperLTL satisfiability by *finite* sets of ultimately periodic traces is Σ_1^0 -complete. The Σ_1^1 -completeness of HyperLTL satisfiability in the general case implies that, in particular, the set of satisfiable HyperLTL sentences is neither recursively enumerable nor co-recursively enumerable. A semi-decision procedure, like the one introduced in [20] for finite sets of ultimately periodic traces, therefore cannot exist in general.

The Σ_1^1 upper bound relies on the fact that every satisfiable HyperLTL formula has a countable model [26]. This allows us to represent these models, and Skolem functions on them, by sets of natural numbers, which are type 1 objects. In this encoding, trace assignments are type 0 objects, as traces in a countable set can be identified by natural numbers. With some more existential type 1 quantification one can then express the existence of a function witnessing that every trace assignment consistent with the Skolem functions satisfies the quantifier-free part of the formula under consideration.

► **Lemma 2.** *HyperLTL satisfiability is in Σ_1^1 .*

Proof. Let φ be a HyperLTL formula, let Φ denote the set of quantifier-free subformulas of φ , and let Π be a trace assignment whose domain contains the variables of φ . The expansion of φ on Π is the function $e_{\varphi, \Pi}: \Phi \times \mathbb{N} \rightarrow \{0, 1\}$ with

$$e_{\varphi, \Pi}(\psi, j) = \begin{cases} 1 & \text{if } \Pi[j, \infty) \models \psi, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

The expansion is completely characterised by the following consistency conditions:

- $e_{\varphi, \Pi}(a_{\pi}, j) = 1$ if and only if $a \in \Pi(\pi)(j)$.
- $e_{\varphi, \Pi}(\neg\psi, j) = 1$ if and only if $e_{\varphi, \Pi}(\psi, j) = 0$.
- $e_{\varphi, \Pi}(\psi_1 \vee \psi_2, j) = 1$ if and only if $e_{\varphi, \Pi}(\psi_1, j) = 1$ or $e_{\varphi, \Pi}(\psi_2, j) = 1$.
- $e_{\varphi, \Pi}(\mathbf{X}\psi, j) = 1$ if and only if $e_{\varphi, \Pi}(\psi, j + 1) = 1$.
- $e_{\varphi, \Pi}(\psi_1 \mathbf{U} \psi_2, j) = 1$ if and only if there is a $j' \geq j$ such that $e_{\varphi, \Pi}(\psi_2, j') = 1$ and $e_{\varphi, \Pi}(\psi_2, j'') = 1$ for all j'' in the range $j \leq j'' < j'$.

Every satisfiable HyperLTL sentence has a countable model [26]. Hence, to prove that the HyperLTL satisfiability problem is in Σ_1^1 , we express, for a given HyperLTL sentence encoded as a natural number, the existence of the following type 1 objects (relying on the fact that there is a bijection between finite sequences over \mathbb{N} and \mathbb{N} itself):

- A countable set of traces over the propositions of φ encoded as a function T from $\mathbb{N} \times \mathbb{N}$ to \mathbb{N} , mapping trace names and positions to (encodings of) subsets of the set of propositions appearing in φ .

- A function S from $\mathbb{N} \times \mathbb{N}^*$ to \mathbb{N} to be interpreted as Skolem functions for the existentially quantified variables of φ , i.e. we map a variable (identified by a natural number) and a trace assignment of the variables preceding it (encoded as a sequence of natural numbers) to a trace name.
- A function E from $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ to \mathbb{N} , where, for a fixed $a \in \mathbb{N}$ encoding a trace assignment Π , the function $x, y \mapsto E(a, x, y)$ is interpreted as the expansion of φ on Π , i.e. x encodes a subformula in Φ and y is a position.

Then, we express the following properties using only type 0 quantification: For every trace assignment of the variables in φ , encoded by $a \in \mathbb{N}$, if a is consistent with the Skolem function encoded by S , then the function $x, y \mapsto E(a, x, y)$ satisfies the consistency conditions characterising the expansion, and we have $E(a, x_0, 0) = 1$, where x_0 is the encoding of the maximal quantifier-free subformula of φ . We leave the tedious, but standard, details to the industrious reader. ◀

Now, we prove hardness.

► **Lemma 3.** *HyperLTL satisfiability is Σ_1^1 -hard.*

Proof. By a reduction from the *recurring tiling problem* which is given as follows. A *tile* is a function $\tau: \{\text{east, west, north, south}\} \rightarrow \mathcal{C}$ that maps directions into a finite set \mathcal{C} of colours. Given a finite set \mathcal{T} of tiles, a *tiling of the positive quadrant* with \mathcal{T} is a function $T: \mathbb{N} \times \mathbb{N} \rightarrow \mathcal{T}$ with the property that:

- if $T(i, j) = \tau_1$ and $T(i + 1, j) = \tau_2$, then $\tau_1(\text{east}) = \tau_2(\text{west})$ and
- if $T(i, j) = \tau_1$ and $T(i, j + 1) = \tau_2$ then $\tau_1(\text{north}) = \tau_2(\text{south})$.

The *recurring tiling problem* is to determine, given a finite set \mathcal{T} of tiles and a designated $\tau_0 \in \mathcal{T}$, whether there is a tiling T of the positive quadrant with \mathcal{T} such that there are infinitely many $j \in \mathbb{N}$ such that $T(0, j) = \tau_0$. This problem is known to be Σ_1^1 -complete [29], so if we reduce it to HyperLTL satisfiability this will establish the desired hardness result.

In our reduction, each x -coordinate in the positive quadrant will be represented by a trace, and each y -coordinate by a point in time.² In order to keep track of which trace represents which x -coordinate, we use one designated atomic proposition x that holds on exactly one time point in each trace: x holds at time i if and only if the trace represents x -coordinate i .

For this purpose, let \mathcal{T} be given, and define the following formulas over $\text{AP} = \{x\} \cup \mathcal{T}$:

- Every trace has exactly one point where x holds:

$$\varphi_1 = \forall \pi. (\neg x_\pi \mathbf{U}(x_\pi \wedge \mathbf{X} \mathbf{G} \neg x_\pi))$$

- For every $i \in \mathbb{N}$, there is a trace with x in the i -th position:

$$\varphi_2 = (\exists \pi. x_\pi) \wedge (\forall \pi_1. \exists \pi_2. \mathbf{F}(x_{\pi_1} \wedge \mathbf{X} x_{\pi_2}))$$

- If two traces represent the same x -coordinate, then they contain the same tiles:

$$\varphi_3 = \forall \pi_1. \forall \pi_2. (\mathbf{F}(x_{\pi_1} \wedge x_{\pi_2}) \rightarrow \mathbf{G}(\bigwedge_{\tau \in \mathcal{T}} (\tau_{\pi_1} \leftrightarrow \tau_{\pi_2})))$$

² Note that this means that if we were to visually represent this construction, traces would be arranged vertically.

47:8 HyperLTL Satisfiability Is Σ_1^1 -Complete, HyperCTL* Satisfiability Is Σ_1^2 -Complete

- Every time point in every trace contains exactly one tile:

$$\varphi_4 = \forall \pi. \mathbf{G} \bigvee_{\tau \in \mathcal{T}} (\tau_\pi \wedge \bigwedge_{\tau' \in \mathcal{T} \setminus \{\tau\}} \neg(\tau')_\pi)$$

- Tiles match vertically:

$$\varphi_5 = \forall \pi. \mathbf{G} \bigvee_{\tau \in \mathcal{T}} (\tau_\pi \wedge \bigvee_{\tau' \in \{\tau' \in \mathcal{T} \mid \tau(\text{north}) = \tau'(\text{south})\}} \mathbf{X}(\tau')_\pi)$$

- Tiles match horizontally:

$$\varphi_6 = \forall \pi_1. \forall \pi_2. (\mathbf{F}(x_{\pi_1} \wedge \mathbf{X}x_{\pi_2}) \rightarrow \mathbf{G} \bigvee_{\tau \in \mathcal{T}} (\tau_{\pi_1} \wedge \bigvee_{\tau' \in \{\tau' \in \mathcal{T} \mid \tau(\text{east}) = \tau'(\text{west})\}} (\tau')_{\pi_2}))$$

- Tile τ_0 occurs infinitely often at x -position 0:

$$\varphi_7 = \exists \pi. (x_\pi \wedge \mathbf{GF} \tau_0)$$

Finally, take $\varphi_{\mathcal{T}} = \bigwedge_{1 \leq i \leq 7} \varphi_i$. Technically $\varphi_{\mathcal{T}}$ is not a HyperLTL formula, since it is not in prenex normal form, but it can be trivially transformed into one. Collectively, subformulas φ_1 – φ_3 are satisfied in exactly those sets of traces that can be interpreted as $\mathbb{N} \times \mathbb{N}$. Subformulas φ_4 – φ_6 then hold if and only if the $\mathbb{N} \times \mathbb{N}$ grid is correctly tiled with \mathcal{T} . Subformula φ_7 , finally, holds if and only if the tiling uses the tile τ_0 infinitely often at x -coordinate 0. Overall, this means $\varphi_{\mathcal{T}}$ is satisfiable if and only if \mathcal{T} can recurrently tile the positive quadrant.

The Σ_1^1 -hardness of HyperLTL satisfiability therefore follows from the Σ_1^1 -hardness of the recurring tiling problem [29]. ◀

The Σ_1^1 -completeness of HyperLTL satisfiability still holds if we restrict to ultimately periodic traces.

► **Theorem 4.** *HyperLTL satisfiability restricted to sets of ultimately periodic traces is Σ_1^1 -complete.*

Proof. The problem of whether there is a tiling of $\{(i, j) \in \mathbb{N}^2 \mid i \geq j\}$, i.e. the part of $\mathbb{N} \times \mathbb{N}$ below the diagonal, such that a designated tile τ_0 occurs on every row, is also Σ_1^1 -complete [29].³ We reduce this problem to HyperLTL satisfiability on ultimately periodic traces.

The reduction is very similar to the one discussed above, with the necessary changes being: (i) every time point beyond x satisfies the special tile “null”, (ii) horizontal and vertical matching are only checked at or before time point x and (iii) for every π_1 there is a π_2 such that π_2 has designated tile τ_0 at the time where π_1 satisfies x (so τ_0 holds at least once in every row).

Membership in Σ_1^1 can be shown similarly to the proof of Lemma 2. So, the problem is Σ_1^1 -complete. ◀

³ The proof in [29] is for the part *above* the diagonal with τ_0 occurring on every column, but that is easily seen to be equivalent.

4 The HyperLTL Quantifier Alternation Hierarchy

The number of quantifier alternations in a formula is a crucial parameter in the complexity of HyperLTL model-checking [25, 35]. A natural question is then to understand *which* properties can be expressed with n quantifier alternations, that is, given a sentence φ , determine if there exists an equivalent one with at most n alternations. In this section, we show that this problem is in fact exactly as hard as the HyperLTL unsatisfiability problem (which asks whether a HyperLTL sentence has no model), and therefore Π_1^1 -complete. Here, Π_1^1 is the co-class of Σ_1^1 , i.e. it contains the complements of the Σ_1^1 sets.

Formally, the HyperLTL quantifier alternation hierarchy is defined as follows. Let φ be a HyperLTL formula. We say that φ is a Σ_0 - or a Π_0 -formula if it is quantifier-free. It is a Σ_n -formula if it is of the form $\varphi = \exists\pi_1 \cdots \exists\pi_k. \psi$ and ψ is a Π_{n-1} -formula. It is a Π_n -formula if it is of the form $\varphi = \forall\pi_1 \cdots \forall\pi_k. \psi$ and ψ is a Σ_{n-1} -formula. We do not require each block of quantifiers to be non-empty, i.e. we may have $k = 0$ and $\varphi = \psi$. By a slight abuse of notation, we also let Σ_n denote the set of hyperproperties definable by a Σ_n -sentence, that is, the set of all $L(\varphi) = \{T \subseteq (2^{\text{AP}})^\omega \mid T \models \varphi\}$ such that φ is a Σ_n -sentence of HyperLTL.

► **Theorem 5** ([35, Corollary 5.6.5]). *The quantifier alternation hierarchy of HyperLTL is strict: for all $n > 0$, $\Sigma_n \subsetneq \Sigma_{n+1}$.*

The strictness of the hierarchy also holds if we restrict our attention to sentences whose models consist of finite sets of traces that end in the suffix \emptyset^ω , i.e. that are essentially finite.

► **Theorem 6.** *For all $n > 0$, there exists a Σ_{n+1} -sentence φ of HyperLTL that is not equivalent to any Σ_n -sentence, and such that for all $T \subseteq (2^{\text{AP}})^\omega$, if $T \models \varphi$ then T contains finitely many traces and $T \subseteq (2^{\text{AP}})^* \emptyset^\omega$.*

This fact is a necessary ingredient for our argument that membership at some fixed level of the quantifier alternation hierarchy is Π_1^1 -hard. It could be derived from a small adaptation of the proof in [35], and we provide an alternative proof in the extended version [27] by exhibiting a connection between the HyperLTL quantifier alternation hierarchy and the quantifier alternation hierarchy for first-order logic over finite words, which is known to be strict [15, 38].

Our goal is to prove the following.

► **Theorem 7.** *Fix $n > 0$. The problem of deciding whether a HyperLTL sentence is equivalent to some Σ_n -sentence is Π_1^1 -complete.*

The easier part is the upper bound, since a corollary of Theorem 1 is that the problem of deciding whether two HyperLTL formulas are equivalent is Π_1^1 -complete. The lower bound is proven by reduction from the HyperLTL unsatisfiability problem. The proof relies on Theorem 6: given a sentence φ , we are going to combine φ with some Σ_{n+1} -sentence φ_{n+1} witnessing the strictness of the hierarchy, to construct a sentence ψ such that φ is unsatisfiable if and only if ψ is equivalent to a Σ_n -sentence. Intuitively, the formula ψ will describe models consisting of the “disjoint union” of a model of φ_{n+1} and a model of φ . Here “disjoint” is to be understood in a strong sense: we split both the set of traces and the time domain into two parts, used respectively to encode the models of φ_{n+1} and those of φ .

To make this more precise, let us introduce some notations. We assume a distinguished symbol $\$ \notin \text{AP}$. We say that a set of traces $T \subseteq (2^{\text{AP} \cup \{\$\}})^\omega$ is *bounded* if there exists $b \in \mathbb{N}$ such that $T \subseteq (2^{\text{AP}})^b \cdot \{\$\}^\omega$.

47:10 HyperLTL Satisfiability Is Σ_1^1 -Complete, HyperCTL* Satisfiability Is Σ_1^2 -Complete

T_ℓ	$\{a, b\}$	$\{a\}$	$\{a\}$	$\{\$\}$	$\{\$\}$	$\{\$\}$	$\{\$\}$	$\{\$\}$	\dots
	$\{b\}$	\emptyset	$\{a\}$	$\{\$\}$	$\{\$\}$	$\{\$\}$	$\{\$\}$	$\{\$\}$	\dots
	$\{\$\}$	$\{\$\}$	$\{\$\}$	$\{a\}$	$\{a\}$	$\{a, b\}$	\emptyset	$\{a\}$	\dots
	$\{\$\}$	$\{\$\}$	$\{\$\}$	$\{b\}$	$\{a\}$	$\{b\}$	$\{a, b\}$	$\{a\}$	\dots
								T_r	

■ **Figure 1** Example of a split set of traces where each row represents a trace and $b = 3$.

► **Lemma 8.** *There exists a Π_1 -sentence φ_{bd} such that for all $T \subseteq (2^{\text{AP} \cup \{\$\}})^\omega$, we have $T \models \varphi_{bd}$ if and only if T is bounded.*

Proof. We let

$$\varphi_{bd} = \forall \pi. \forall \pi'. (\neg \$_\pi \mathbf{U} \mathbf{G} \$_\pi) \wedge \bigwedge_{a \in \text{AP}} \mathbf{G}(\neg(a_\pi \wedge \$_\pi)) \wedge \mathbf{F}(\neg \$_\pi \wedge \neg \$_{\pi'} \wedge \mathbf{X} \$_\pi \wedge \mathbf{X} \$_{\pi'}).$$

The conjunct $(\neg \$_\pi \mathbf{U} \mathbf{G} \$_\pi) \wedge \bigwedge_{a \in \text{AP}} \mathbf{G}(\neg(a_\pi \wedge \$_\pi))$ ensures that every trace is in $(2^{\text{AP}})^* \cdot \{\$\}^\omega$, while $\mathbf{F}(\neg \$_\pi \wedge \neg \$_{\pi'} \wedge \mathbf{X} \$_\pi \wedge \mathbf{X} \$_{\pi'})$ ensures that the $\$\text{'s}$ in any two traces π and π' start at the same position. ◀

We say that T is *split* if there exist $b \in \mathbb{N}$ and T_1, T_2 such that $T = T_1 \uplus T_2$, $T_1 \subseteq (2^{\text{AP}})^b \cdot \{\$\}^\omega$, and $T_2 \subseteq \{\$\}^b \cdot (2^{\text{AP}})^\omega$. Note that b is unique here. Hence, we define the left and right part of T as $T_\ell = T_1$ and $T_r = \{t \in (2^{\text{AP}})^\omega \mid \{\$\}^b \cdot t \in T_2\}$, respectively (see Figure 1).

It is easy to combine HyperLTL specifications for the left and right part of a split model into one global formula.

► **Lemma 9.** *For all HyperLTL sentences φ_ℓ, φ_r , one can construct a sentence ψ such that for all split $T \subseteq (2^{\text{AP} \cup \{\$\}})^\omega$, it holds that $T_\ell \models \varphi_\ell$ and $T_r \models \varphi_r$ if and only if $T \models \psi$.*

Proof of Lemma 9. Let $\widehat{\varphi}_r$ denote the formula obtained from φ_r by replacing:

- every existential quantification $\exists \pi. \varphi$ with $\exists \pi. ((\mathbf{F} \mathbf{G} \neg \$_\pi) \wedge \varphi)$;
 - every universal quantification $\forall \pi. \varphi$ with $\forall \pi. ((\mathbf{F} \mathbf{G} \neg \$_\pi) \rightarrow \varphi)$;
 - the quantifier-free part φ of φ_r with $\$_\pi \mathbf{U}(\neg \$_\pi \wedge \varphi)$, where π is some free variable in φ .
- Here, the first two replacements restrict quantification to traces in the right part while the last one requires the formula to hold at the first position of the right part. We define $\widehat{\varphi}_\ell$ by similarly relativizing quantifications in φ_ℓ . The formula $\widehat{\varphi}_\ell \wedge \widehat{\varphi}_r$ can then be put back into prenex normal form to define ψ . ◀

Conversely, any HyperLTL formula that only has split models can be decomposed into a Boolean combination of formulas that only talk about the left or right part of the model. This is formalised in the lemma below.

► **Lemma 10.** *For all HyperLTL Σ_n -sentences φ there exists a finite family $(\varphi_\ell^i, \varphi_r^i)_i$ of Σ_n -sentences such that for all split $T \subseteq (2^{\text{AP} \cup \{\$\}})^\omega$: $T \models \varphi$ if and only if there is an i with $T_\ell \models \varphi_\ell^i$ and $T_r \models \varphi_r^i$.*

We are now ready to prove Theorem 7.

Proof of Theorem 7. The upper bound is an easy consequence of Theorem 1: Given a HyperLTL sentence φ , we express the existence of a Σ_n -sentence ψ using first-order quantification and encode equivalence of ψ and φ via the formula $(\neg \varphi \wedge \psi) \vee (\varphi \wedge \neg \psi)$, which is unsatisfiable if and only if φ and ψ are equivalent. Altogether, this shows membership in Π_1^1 , as Π_1^1 is closed under existential first-order quantification (see, e.g. [30, Page 82]).

We prove the lower bound by reduction from the unsatisfiability problem for HyperLTL. So given a HyperLTL sentence φ , we want to construct ψ such that φ is unsatisfiable if and only if ψ is equivalent to a Σ_n -sentence.

We first consider the case $n > 1$. Fix a Σ_{n+1} -sentence φ_{n+1} that is not equivalent to any Σ_n -sentence, and such that every model of φ_{n+1} is bounded. The existence of such a formula is a consequence of Theorem 6. By Lemma 9, there exists a computable ψ such that for all split models T , we have $T \models \psi$ if and only if $T_\ell \models \varphi_{n+1}$ and $T_r \models \varphi$.

First, it is clear that if φ is unsatisfiable, then ψ is unsatisfiable as well, and thus equivalent to $\exists\pi. a_\pi \wedge \neg a_\pi$, which is a Σ_n -sentence since $n \geq 1$.

Conversely, suppose towards a contradiction that φ is satisfiable and that ψ is equivalent to some Σ_n -sentence. Let $(\psi_\ell^i, \psi_r^i)_i$ be the finite family of Σ_n -sentences given by Lemma 10 for ψ . Fix a model T_φ of φ . For a bounded T , we let \bar{T} denote the unique split set of traces such that $\bar{T}_\ell = T$ and $\bar{T}_r = T_\varphi$. For all T , we then have $T \models \varphi_{n+1}$ if and only if T is bounded and $\bar{T} \models \psi$. Recall that the set of bounded models can be defined by a Π_1 -sentence φ_{bd} (Lemma 8), which is also a Σ_n -sentence since $n > 1$. We then have $T \models \varphi_{n+1}$ if and only if $T \models \varphi_{bd}$ and there exists i such that $T \models \psi_\ell^i$ and $T_\varphi \models \psi_r^i$. So φ_{n+1} is equivalent to

$$\varphi_{bd} \wedge \bigvee_{i \text{ with } T_\varphi \models \psi_r^i} \psi_\ell^i,$$

which, since Σ_n -sentences are closed (up to logical equivalence) under conjunction and disjunction, is equivalent to a Σ_n -sentence. This contradicts the definition of φ_{n+1} .

We are left with the case $n = 1$. Similarly, we construct ψ such that φ is unsatisfiable if and only if ψ is unsatisfiable, and if and only if ψ is equivalent to a Σ_1 -sentence. However, we do not need to use bounded or split models here. Every satisfiable Σ_1 -sentence has a model with finitely many traces. Therefore, a simple way to construct ψ so that it is not equivalent to any Σ_1 -sentence (unless it is unsatisfiable) is to ensure that every model of ψ contains infinitely many traces.

Let $x \notin \text{AP}$, and $T_\omega = \{\emptyset^n \{x\} \emptyset^\omega \mid n \in \mathbb{N}\}$. As seen in the proof of Lemma 3, T_ω is definable in HyperLTL: There is a sentence φ_ω such that $T \subseteq (2^{\text{AP} \cup \{x\}})^\omega$ is a model of φ_ω if and only if $T = T_\omega$. By relativising quantifiers in φ_ω and φ to traces with or without the atomic proposition x , one can construct a HyperLTL sentence ψ such that $T \models \psi$ if and only if $T_\omega \subseteq T$ and $T \setminus T_\omega \models \varphi$.

Again, if φ is unsatisfiable then ψ is unsatisfiable and therefore equivalent to $\exists\pi. a_\pi \wedge \neg a_\pi$, a Σ_1 -sentence. Conversely, all models of ψ contain infinitely many traces and therefore, if ψ is equivalent to a Σ_1 -sentence then it is unsatisfiable, and so is φ . ◀

5 HyperCTL* satisfiability is Σ_1^2 -complete

Here, we consider the HyperCTL* satisfiability problem: given a HyperLTL sentence, determine whether it has a model \mathcal{T} (of arbitrary size). We prove that it is much harder than HyperLTL satisfiability. As a key step of the proof, we also prove that every satisfiable sentence admits a model of cardinality at most \mathfrak{c} (the cardinality of the continuum), and conversely, we exhibit a satisfiable sentence whose models are all of cardinality at least \mathfrak{c} .

► **Theorem 11.** *HyperCTL* satisfiability is Σ_1^2 -complete.*

On the other hand, HyperCTL* satisfiability restricted to finite transition systems is Σ_1^0 -complete. The upper bound follows from HyperCTL* model checking being decidable [12] (therefore, the problem is recursively enumerable and thus in Σ_1^0) while the matching lower bound is inherited from HyperLTL [19].

47:12 HyperLTL Satisfiability Is Σ_1^1 -Complete, HyperCTL* Satisfiability Is Σ_1^2 -Complete

Upper bound. We begin by proving membership in Σ_1^2 . The first step is to obtain a bound on the size of minimal models of satisfiable HyperCTL* sentences. For this, we use an argument based on Skolem functions, which is a transfinite generalisation of the proof that all satisfiable HyperLTL sentences have a countable model [26].

In the following, we use ω and ω_1 to denote the first infinite and the first uncountable ordinal, respectively, and write \aleph_0 and \aleph_1 for their cardinality.

► **Lemma 12.** *Each satisfiable HyperCTL* sentence φ has a model of size at most \mathfrak{c} .*

Proof sketch. Suppose φ has a model \mathcal{T} of arbitrary size, and fix Skolem functions witnessing this satisfaction. We then create a transfinite sequence of transition systems \mathcal{T}_α . We start by taking \mathcal{T}_0 to be any single path from \mathcal{T} starting in the initial vertex, and obtain $\mathcal{T}_{\alpha+1}$ by adding to \mathcal{T}_α all vertices and edges of the paths that are the outputs of the Skolem functions when restricted to inputs from \mathcal{T}_α . If α is a limit ordinal we take \mathcal{T}_α to be the union of all previous transition systems.

This sequence does not necessarily stabilise at ω , since \mathcal{T}_ω may contain a path ρ such that $\rho(i)$ was introduced in \mathcal{T}_i . This would result in \mathcal{T}_ω containing a path that was not present in any earlier model \mathcal{T}_i with $i < \omega$, and therefore we could have $\mathcal{T}_{\omega+1} \neq \mathcal{T}_\omega$.

The sequence does stabilise at ω_1 , however. This is because every path ρ contains only countably many vertices, so if every element $\rho(i)$ of ρ is introduced at some countable α_i , then there is a countable α such that all of ρ is included in \mathcal{T}_α . It follows that \mathcal{T}_{ω_1} does not contain any “new” paths that were not already in some \mathcal{T}_α with $\alpha < \omega_1$, and therefore the Skolem function f does not generate any “new” outputs either.

In each step of the construction at most \mathfrak{c} new vertices are added, so \mathcal{T}_{ω_1} contains at most \mathfrak{c} vertices. Furthermore, because \mathcal{T}_{ω_1} is closed under the Skolem functions, the satisfaction of φ in \mathcal{T} implies its satisfaction in \mathcal{T}_{ω_1} . ◀

With the upper bound at hand, we can place HyperCTL* satisfiability in Σ_1^2 , as the existence of a model of size \mathfrak{c} can be captured by quantification over type 2 objects.

► **Lemma 13.** *HyperCTL* satisfiability is in Σ_1^2 .*

Proof. As in the proof of Theorem 1. Because every HyperCTL* formula is satisfied in a model of size at most \mathfrak{c} , these models can be represented by objects of type 2. Checking whether a formula is satisfied in a transition system is equivalent to the existence of a winning strategy for Verifier in the induced model checking game. Such a strategy is again a type 2 object, which is existentially quantified. Finally, whether it is winning can be expressed by quantification over individual elements and paths, which are objects of types 0 and 1. Checking the satisfiability of a HyperCTL* formula φ therefore amounts to existential third-order quantification (to choose a model and a winning strategy) followed by a second-order formula to verify that φ holds on the model. Hence HyperCTL* satisfiability is in Σ_1^2 .

Formally, we encode the existence of a winning strategy for Verifier in the HyperCTL* model checking game $\mathcal{G}(\mathcal{T}, \varphi)$ induced by a transition system \mathcal{T} and a HyperCTL* formula φ . This game is played between Verifier and Falsifier, one of them aiming to prove that $\mathcal{T} \models \varphi$ and the other aiming to prove $\mathcal{T} \not\models \varphi$. It is played in a graph whose positions correspond to subformulas which they want to check (and suitable path assignments of the free variables): each vertex (say, representing a subformula ψ) belongs to one of the players who has to pick a successor, which represents a subformula of ψ . A play ends at an atomic proposition, at which point the winner can be determined.

Formally, a vertex of the game is of the form (Π, ψ, b) where Π is a path assignment, ψ is a subformula of φ , and $b \in \{0, 1\}$ is a flag used to count the number of negations encountered along the play; the initial vertex is $(\Pi_\emptyset, \varphi, 0)$. Furthermore, for until-subformulas ψ , we need auxiliary vertices of the form (Π, ψ, b, j) with $j \in \mathbb{N}$. The vertices of Verifier are

- of the form $(\Pi, \psi, 0)$ with $\psi = \psi_1 \vee \psi_2$, $\psi = \psi_1 \mathbf{U} \psi_2$, or $\psi = \exists\pi. \psi'$,
- of the form $(\Pi, \forall\pi. \psi', 1)$, or
- of the form $(\Pi, \psi_1 \mathbf{U} \psi_2, 1, j)$.

The moves of the game are defined as follows:

- A vertex (Π, a_π, b) is terminal. It is winning for Verifier if $b = 0$ and $a \in \lambda(\Pi(\pi)(0))$ or if $b = 1$ and $a \notin \lambda(\Pi(\pi)(0))$, where λ is the labelling function of \mathcal{T} .
- A vertex $(\Pi, \neg\psi, b)$ has a unique successor $(\Pi, \psi, b + 1 \bmod 2)$.
- A vertex $(\Pi, \psi_1 \vee \psi_2, b)$ has two successors of the form (Π, ψ_i, b) for $i \in \{1, 2\}$.
- A vertex $(\Pi, \mathbf{X}\psi, b)$ has a unique successor $(\Pi[1, \infty), \psi, b)$.
- A vertex $(\Pi, \psi_1 \mathbf{U} \psi_2, b)$ has a successor $(\Pi, \psi_1 \mathbf{U} \psi_2, b, j)$ for every $j \in \mathbb{N}$.
- A vertex $(\Pi, \psi_1 \mathbf{U} \psi_2, b, j)$ has the successor $(\Pi[j, \infty), \psi_2, b)$ as well as successors $(\Pi[j', \infty), \psi_1, b)$ for every $0 \leq j' < j$.
- A vertex $(\Pi, \exists\pi. \psi, b)$ has successors $(\Pi[\pi \mapsto \rho], \psi, b)$ for every path ρ of \mathcal{T} starting in $\text{rnt}(\Pi)$.
- A vertex $(\Pi, \forall\pi. \psi, b)$ has successors $(\Pi[\pi \mapsto \rho], \psi, b)$ for every path ρ of \mathcal{T} starting in $\text{rnt}(\Pi)$.

A play of the model checking game is a finite path through the graph, starting at the initial vertex and ending at a terminal vertex. It is winning for Verifier if the terminal vertex is winning for her. Note that the length of a play is bounded by $2d$, where d is the depth⁴ of φ , as the formula is simplified during each move.

A strategy σ for Verifier is a function mapping each of her vertices v to some successor of v . A play $v_0 \cdots v_k$ is consistent with σ , if $v_{k'+1} = \sigma(v_{k'})$ for every $0 \leq k' < k$ such that $v_{k'}$ is a vertex of Verifier. A straightforward induction shows that Verifier has a winning strategy for $\mathcal{G}(\mathcal{T}, \varphi)$ if and only if $\mathcal{T} \models \varphi$.

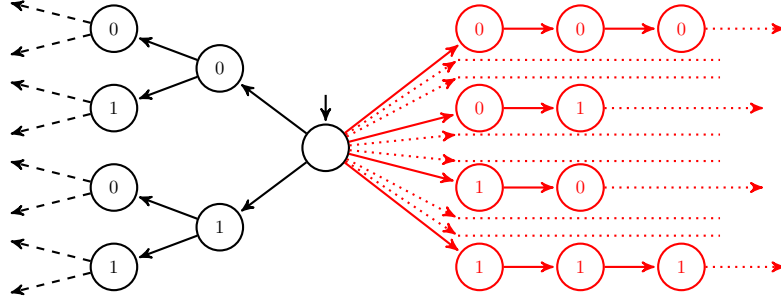
Recall that every satisfiable HyperCTL* sentence has a model of cardinality \mathfrak{c} (Lemma 12). Thus, to place HyperCTL* satisfiability in Σ_1^2 , we express, for a given natural number encoding a HyperCTL* formula φ , the existence of the following type 2 objects (using suitable encodings):

- A transition system \mathcal{T} of cardinality \mathfrak{c} .
- A function σ from V to V , where V is the set of vertices of $\mathcal{G}(\mathcal{T}, \varphi)$. Note that a single vertex of V is a type 1 object.

Then, we express that σ is a strategy for Verifier, which is easily expressible using quantification over type 1 objects. Thus, it remains to express that σ is winning by stating that every play (a sequence of type 1 objects of bounded length) that is consistent with σ ends in a terminal vertex that is winning for Verifier. Again, we leave the tedious, but standard, details to the reader. ◀

Lower bound. We first describe a satisfiable HyperCTL* sentence $\varphi_{\mathfrak{c}}$ that does not have any model of cardinality less than \mathfrak{c} (more precisely, the initial vertex must have uncountably many successors), thus matching the upper bound from Lemma 12. We construct $\varphi_{\mathfrak{c}}$ with one particular model $\mathcal{T}_{\mathfrak{c}}$ in mind, defined below, though it also admits other models.

⁴ The depth is the maximal nesting of quantifiers, Boolean connectives, and temporal operators.



■ **Figure 2** A depiction of \mathcal{T}_c . Vertices in black (on the left including the initial vertex) are labelled by **fbt**, those in red (on the right, excluding the initial vertex) are labelled by **set**.

The idea is that we want all possible subsets of $A \subseteq \mathbb{N}$ to be represented in \mathcal{T}_c in the form of paths ρ_A such that $\rho_A(i)$ is labelled by 1 if $i \in A$, and by 0 otherwise. By ensuring that the first vertices of these paths are pairwise distinct, we obtain the desired lower bound on the cardinality. We express this in HyperCTL* as follows: First, we express that there is a part of the model (labelled by **fbt**) where every reachable vertex has two successors, one labelled with 0 and one labelled with 1, i.e. the unravelling of this part contains the full binary tree. Thus, this part has a path ρ_A as above for every subset A , but their initial vertices are not necessarily distinct. Hence, we also express that there is another part (labelled by **set**) that contains a copy of each path in the **fbt**-part, and that these paths indeed start at distinct successors of the initial vertex.

We let $\mathcal{T}_c = (V_c, E_c, t_\varepsilon, \lambda_c)$ (see Figure 2), where

$$V_c = \{t_u \mid u \in \{0, 1\}^*\} \cup \{s_A^i \mid i \in \mathbb{N} \wedge A \subseteq \mathbb{N}\}$$

$$\lambda_c(t_\varepsilon) = \{\mathbf{fbt}\} \quad \lambda_c(t_{u.0}) = \{\mathbf{fbt}, 0\} \quad \lambda_c(t_{u.1}) = \{\mathbf{fbt}, 1\} \quad \lambda_c(s_A^i) = \begin{cases} \{\mathbf{set}, 0\} & \text{if } i \notin A \\ \{\mathbf{set}, 1\} & \text{if } i \in A \end{cases}$$

$$E_c = \{(t_u, t_{u0}), (t_u, t_{u1}) \mid u \in \{0, 1\}^*\} \cup \{(t_\varepsilon, s_A^0) \mid A \subseteq \mathbb{N}\} \cup \{(s_A^i, s_A^{i+1}) \mid A \subseteq \mathbb{N}, i \in \mathbb{N}\}.$$

► **Lemma 14.** *There is a satisfiable HyperCTL* sentence φ_c that has only models of cardinality at least c .*

Proof. The formula φ_c is defined as the conjunction of the formulas below:

1. The label of the initial vertex is $\{\mathbf{fbt}\}$ and the labels of non-initial vertices are $\{\mathbf{fbt}, 0\}$, $\{\mathbf{fbt}, 1\}$, $\{\mathbf{set}, 0\}$, or $\{\mathbf{set}, 1\}$:

$$\forall \pi. (\mathbf{fbt}_\pi \wedge \neg 0_\pi \wedge \neg 1_\pi \wedge \neg \mathbf{set}_\pi) \wedge \mathbf{XG}((\mathbf{set}_\pi \leftrightarrow \neg \mathbf{fbt}_\pi) \wedge (0_\pi \leftrightarrow \neg 1_\pi))$$

2. All **fbt**-labelled vertices have a successor with label $\{\mathbf{fbt}, 0\}$ and one with label $\{\mathbf{fbt}, 1\}$, and all **fbt**-labelled vertices that are additionally labelled by 0 or 1 have no **set**-labelled successor:

$$\forall \pi. \mathbf{G}(\mathbf{fbt}_\pi \rightarrow ((\exists \pi_0. \mathbf{X}(\mathbf{fbt}_{\pi_0} \wedge 0_{\pi_0})) \wedge (\exists \pi_1. \mathbf{X}(\mathbf{fbt}_{\pi_1} \wedge 1_{\pi_1}))) \wedge ((0_\pi \vee 1_\pi) \rightarrow \forall \pi'. \mathbf{X} \mathbf{fbt}_{\pi'}))$$

3. For every path of **fbt**-labelled vertices starting at a successor of the initial vertex, there is a path of **set**-labelled vertices (also starting at a successor of the initial vertex) with the same $\{0, 1\}$ labelling:

$$\forall \pi. ((\mathbf{X} \mathbf{fbt}_\pi) \rightarrow \exists \pi'. \mathbf{X}(\mathbf{set}_{\pi'} \wedge \mathbf{G}(0_\pi \leftrightarrow 0_{\pi'})))$$

4. Any two paths starting in the same **set**-labelled vertex have the same sequence of labels:

$$\forall \pi. \mathbf{G}(\mathbf{set}_\pi \rightarrow \forall \pi'. \mathbf{G}(0_\pi \leftrightarrow 0_{\pi'})).$$

It is easy to check that $\mathcal{T}_c \models \varphi_c$. Note however that it is not the only model of φ_c : for instance, some paths may be duplicated, or merged after some steps if their label sequences share a common suffix. So, consider an arbitrary transition system $\mathcal{T} = (V, E, v_I, \lambda)$ such that $\mathcal{T} \models \varphi_c$. By condition 2, for every set $A \subseteq \mathbb{N}$, there is a path ρ_A starting at a successor of v_I such that $\lambda(\rho_A(i)) = \{\mathbf{fbt}, 1\}$ if $i \in A$ and $\lambda(\rho_A(i)) = \{\mathbf{fbt}, 0\}$ if $i \notin A$. Condition 3 implies that there is also a **set**-labelled path ρ'_A such that ρ'_A starts at a successor of v_I , and has the same $\{0, 1\}$ labelling as ρ_A . Finally, by condition 4, if $A \neq B$ then $\rho'_A(0) \neq \rho'_B(0)$. ◀

Before moving to the proof that HyperCTL* satisfiability is Σ_1^2 -hard, we introduce one last auxiliary formula that will be used in the reduction, showing that addition and multiplication can be defined in HyperCTL*, and in fact even in HyperLTL, as follows: Let $\text{AP} = \{\mathbf{arg1}, \mathbf{arg2}, \mathbf{res}, \mathbf{add}, \mathbf{mult}\}$ and let $T_{(+, \cdot)}$ be the set of all traces $t \in (2^{\text{AP}})^\omega$ such that

- there are unique $n_1, n_2, n_3 \in \mathbb{N}$ with $\mathbf{arg1} \in t(n_1)$, $\mathbf{arg2} \in t(n_2)$, and $\mathbf{res} \in t(n_3)$, and
- either $\mathbf{add} \in t(n)$ for all n and $n_1 + n_2 = n_3$, or $\mathbf{mult} \in t(n)$ for all n and $n_1 \cdot n_2 = n_3$.

► **Lemma 15.** *There is a HyperLTL sentence $\varphi_{(+, \cdot)}$ which has $T_{(+, \cdot)}$ as unique model.*

To establish Σ_1^2 -hardness, we give an encoding of formulas of existential third-order arithmetic into HyperCTL*. As explained in Section 2, we can (and do for the remainder of the section) assume that first-order (type 0) variables range over natural numbers, second-order (type 1) variables range over sets of natural numbers, and third-order (type 2) variables range over sets of sets of natural numbers.

► **Lemma 16.** *Suppose $\varphi = \exists x_1 \dots \exists x_n. \psi$, where x_1, \dots, x_n are third-order variables, and ψ is a formula of second-order arithmetic. One can construct a HyperCTL* formula φ' such that $(\mathbb{N}, 0, 1, +, \cdot, <, \in)$ is a model of φ if and only if φ' is satisfiable.*

Proof. The idea of the proof is as follows. We represent sets of natural numbers as infinite paths with labels in $\{0, 1\}$, so that quantification over sets of natural numbers in ψ can be replaced by HyperCTL* path quantification. First-order quantification is handled in the same way, but using paths where exactly one vertex is labelled 1. In particular we encode first- and second-order variables x of φ as path variables π_x of φ' . For this to work, we need to make sure that every possible set has a path representative in the transition system (possibly several isomorphic ones). This is where formula φ_c defined in Lemma 14 is used. For arithmetical operations, we rely on the formula $\varphi_{(+, \cdot)}$ from Lemma 15. Finally, we associate with every existentially quantified third-order variable x_i an atomic proposition a_i , so that for a second-order variable y , the atomic formula $y \in x_i$ is interpreted as the atomic proposition a_i being true on π_y . This is all explained in more details below.

Let $\text{AP} = \{a_1, \dots, a_n, 0, 1, \mathbf{set}, \mathbf{fbt}, \mathbf{arg1}, \mathbf{arg2}, \mathbf{res}, \mathbf{mult}, \mathbf{add}\}$. Given an interpretation $\nu : \{x_1, \dots, x_n\} \rightarrow 2^{(2^{\mathbb{N}})}$ of the third-order variables of φ , we denote by \mathcal{T}_ν the transition system over AP obtained as follows: We start from \mathcal{T}_c , and extend it with an $\{a_1, \dots, a_n\}$ -labelling by setting $a_i \in \lambda(\rho_A(0))$ if $A \in \nu(x_i)$; then, we add to this transition system all traces in $T_{(+, \cdot)}$ as disjoint paths below the initial vertex.

From the formulas φ_c and $\varphi_{(+, \cdot)}$ defined in Lemmas 14 and 15, it is not difficult to construct a formula $\varphi_{(c, +, \cdot)}$ such that:

- For all $\nu : \{x_1, \dots, x_n\} \rightarrow 2^{(2^{\mathbb{N}})}$, the transition system \mathcal{T}_ν is a model of $\varphi_{(c, +, \cdot)}$.
- Conversely, in any model $\mathcal{T} = (V, E, v_I, \lambda)$ of $\varphi_{(c, +, \cdot)}$, the following conditions are satisfied:

47:16 HyperLTL Satisfiability Is Σ_1^1 -Complete, HyperCTL* Satisfiability Is Σ_1^2 -Complete

1. For every path ρ starting at a **set**-labelled successor of the initial vertex v_I , the vertex $\rho(0)$ has a label of the form $\lambda(\rho(0)) = \{\mathbf{set}, b\} \cup \ell$ with $b \in \{0, 1\}$ and $\ell \subseteq \{a_1, \dots, a_n\}$, and every vertex $\rho(i)$ with $i > 0$ has a label $\lambda(\rho(i)) = \{\mathbf{set}, 0\}$ or $\lambda(\rho(i)) = \{\mathbf{set}, 1\}$.
2. For every $A \subseteq \mathbb{N}$, there exists a **set**-labelled path ρ_A starting at a successor of v_I such that $1 \in \lambda(\rho_A(i))$ if $i \in A$, and $0 \in \lambda(\rho_A(i))$ if $i \notin A$. Moreover, all such paths have the same $\{a_1, \dots, a_n\}$ labelling; this can be expressed by the formula

$$\forall \pi. \forall \pi'. \mathbf{X} \left(\mathbf{G}(\mathbf{set}_\pi \wedge \mathbf{set}_{\pi'} \wedge (1_\pi \leftrightarrow 1_{\pi'})) \rightarrow \bigwedge_{a \in \{a_1, \dots, a_n\}} a_\pi \leftrightarrow a_{\pi'} \right).$$

3. For every path ρ starting at an **add**- or **mult**-labelled successor of the initial vertex, the label sequence $\lambda(\rho(0))\lambda(\rho(1)) \dots$ of ρ is in $T_{(+, \cdot)}$.
4. Conversely, for every trace $t \in T_{(+, \cdot)}$, there exists a path ρ starting at a successor of the initial vertex such that $\lambda(\rho(0))\lambda(\rho(1)) \dots = t$.

We then let $\varphi' = \varphi_{(\mathbf{c}, +, \cdot)} \wedge \exists \pi_0. \exists \pi_1. \mathbf{X}(1_{\pi_0} \wedge \mathbf{X} \mathbf{G} 0_{\pi_0} \wedge 0_{\pi_1} \wedge \mathbf{X} 1_{\pi_1} \wedge \mathbf{X} \mathbf{X} \mathbf{G} 0_{\pi_1}) \wedge h(\psi)$, where π_0 and π_1 are used to encode the constants 0 and 1, and $h(\psi)$ is defined inductively from the second-order body ψ of φ as follows:

- $h(\psi_1 \vee \psi_2) = h(\psi_1) \vee h(\psi_2)$ and $h(\neg \psi_1) = \neg h(\psi_1)$.
- If x ranges over sets of natural numbers, $h(\exists x. \psi_1) = \exists \pi_x. ((\mathbf{X} \mathbf{set}_{\pi_x}) \wedge h(\psi_1))$, and $h(\forall x. \psi_1) = \forall \pi_x. ((\mathbf{X} \mathbf{set}_{\pi_x}) \rightarrow h(\psi_1))$.
- If x ranges over natural numbers, $h(\exists x. \psi_1) = \exists \pi_x. ((\mathbf{X} \mathbf{set}_{\pi_x}) \wedge \mathbf{X}(0_{\pi_x} \mathbf{U}(1_{\pi_x} \wedge \mathbf{X} \mathbf{G} 0_{\pi_x}))) \wedge h(\psi_1)$, and $h(\forall x. \psi_1) = \forall \pi_x. ((\mathbf{X} \mathbf{set}_{\pi_x}) \wedge \mathbf{X}(0_{\pi_x} \mathbf{U}(1_{\pi_x} \wedge \mathbf{X} \mathbf{G} 0_{\pi_x}))) \rightarrow h(\psi_1)$.
- If y ranges over sets of natural numbers, $h(y \in x_i) = \mathbf{X}(a_i)_{\pi_y}$.
- If x ranges over natural numbers and y over sets of natural numbers, $h(x \in y) = \mathbf{F}(1_{\pi_x} \wedge 1_{\pi_y})$.
- $h(x < y) = \mathbf{F}(1_{\pi_x} \wedge \mathbf{X} \mathbf{F} 1_{\pi_y})$.
- $h(x \cdot y = z) = \exists \pi. (\mathbf{X} \mathbf{add}_\pi) \wedge \mathbf{F}(\mathbf{arg}1_{\pi} \wedge 1_{\pi_x}) \wedge \mathbf{F}(\mathbf{arg}2_{\pi} \wedge 1_{\pi_y}) \wedge \mathbf{F}(\mathbf{res}_\pi \wedge 1_{\pi_z})$, and $h(x + y = z) = \exists \pi. (\mathbf{X} \mathbf{mult}_\pi) \wedge \mathbf{F}(\mathbf{arg}1_{\pi} \wedge 1_{\pi_x}) \wedge \mathbf{F}(\mathbf{arg}2_{\pi} \wedge 1_{\pi_y}) \wedge \mathbf{F}(\mathbf{res}_\pi \wedge 1_{\pi_z})$.

If ψ is true under some interpretation ν of x_1, \dots, x_n as sets of sets of natural numbers, then the transition system \mathcal{T}_ν defined above is a model of φ' . Conversely, if $\mathcal{T} \models \varphi'$ for some transition system \mathcal{T} , then for all sets $A \subseteq \mathbb{N}$ there is a path ρ_A matching A in \mathcal{T} , and all such paths have the same $\{a_1, \dots, a_n\}$ -labelling, so we can define an interpretation ν of x_1, \dots, x_n by taking $A \in \nu(x_i)$ if and only if $a_i \in \lambda(\rho_A(0))$. Under this interpretation ψ holds, and thus φ is true. \blacktriangleleft

► **Lemma 17.** *HyperCTL* satisfiability is Σ_1^2 -hard.*

Proof. Let N be a Σ_1^2 set, i.e. $N = \{x \in \mathbb{N} \mid \exists x_0 \dots \exists x_k. \psi(x, x_0, \dots, x_k)\}$ for some second-order arithmetic formula ψ with existentially quantified third-order variables x_i . For every $n \in \mathbb{N}$, we define a sentence

$$\varphi_n = \exists x_0 \dots \exists x_k. (\exists x. x = \underbrace{0+1+1+\dots+1}_{n \text{ times}} \wedge \psi(x, x_0, \dots, x_k)).$$

Then φ_n is true if and only if $n \in N$. Combining this with Lemma 16, we obtain a computable function that maps any $n \in \mathbb{N}$ to a HyperCTL* formula φ'_n such that $n \in N$ if and only if φ'_n is satisfiable. \blacktriangleleft

6 Conclusion

In this work, we have settled the complexity of the satisfiability problems for HyperLTL and HyperCTL*. In both cases, we significantly increased the lower bounds, i.e. from Σ_1^0 and Σ_1^1 to Σ_1^1 and Σ_1^2 , respectively, and presented the first upper bounds, which are tight in both cases. Along the way, we also determined the complexity of restricted variants, e.g. HyperLTL satisfiability restricted to ultimately periodic traces (or, equivalently, to finite traces) is still Σ_1^1 -complete while HyperCTL* satisfiability restricted to finite transition systems is Σ_1^0 -complete. As a key step in this proof, we showed a tight bound of c on the size of minimal models for satisfiable HyperCTL* sentences. Finally, we also show that deciding membership in any level of the HyperLTL quantifier alternation hierarchy is Π_1^1 -complete.


References

- 1 Erika Ábrahám, Ezio Bartocci, Borzoo Bonakdarpour, and Oyendrila Dobe. Probabilistic hyperproperties with nondeterminism. In Dang Van Hung and Oleg Sokolsky, editors, *ATVA 2020*, volume 12302 of *LNCS*, pages 518–534. Springer, 2020. doi:10.1007/978-3-030-59152-6_29.
- 2 Erika Ábrahám and Borzoo Bonakdarpour. HyperPCTL: A temporal logic for probabilistic hyperproperties. In Annabelle McIver and András Horváth, editors, *QEST 2018*, volume 11024 of *LNCS*, pages 20–35. Springer, 2018. doi:10.1007/978-3-319-99154-2_2.
- 3 Shreya Agrawal and Borzoo Bonakdarpour. Runtime verification of k-safety hyperproperties in HyperLTL. In *CSF 2016*, pages 239–252. IEEE Computer Society, 2016. doi:10.1109/CSF.2016.24.
- 4 Gilles Barthe, Pedro R. D’Argenio, Bernd Finkbeiner, and Holger Hermanns. Facets of software doping. In Tiziana Margaria and Bernhard Steffen, editors, *ISoLA 2016, Proceedings, Part II*, volume 9953 of *LNCS*, pages 601–608, 2016. doi:10.1007/978-3-319-47169-3_46.
- 5 Ezio Bartocci, Thomas Ferrère, Thomas A. Henzinger, Dejan Nickovic, and Ana Oliveira da Costa. Flavours of sequential information flow. *arXiv*, 2021. arXiv:2105.02013.
- 6 Jan Baumeister, Norine Coenen, Borzoo Bonakdarpour, Bernd Finkbeiner, and César Sánchez. A temporal logic for asynchronous hyperproperties. *arXiv*, 2021. arXiv:2104.14025.
- 7 Béatrice Bérard, Stefan Haar, and Loïc Hélouët. Hyper partial order logic. In Sumit Ganguly and Paritosh K. Pandya, editors, *FSTTCS 2018*, volume 122 of *LIPIcs*, pages 20:1–20:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.20.
- 8 Borzoo Bonakdarpour and Bernd Finkbeiner. Runtime verification for HyperLTL. In Yliès Falcone and César Sánchez, editors, *RV 2016*, volume 10012 of *LNCS*, pages 41–45. Springer, 2016. doi:10.1007/978-3-319-46982-9_4.
- 9 Borzoo Bonakdarpour and Bernd Finkbeiner. Controller synthesis for hyperproperties. In *CSF 2020*, pages 366–379. IEEE, 2020. doi:10.1109/CSF49147.2020.00033.
- 10 Laura Bozzelli, Adriano Peron, and Cesar Sanchez. Asynchronous extensions of hyperltl, 2021. arXiv:2104.12886.
- 11 Noel Brett, Umair Siddique, and Borzoo Bonakdarpour. Rewriting-based runtime verification for alternation-free HyperLTL. In Axel Legay and Tiziana Margaria, editors, *TACAS 2017, Part II*, volume 10206 of *LNCS*, pages 77–93, 2017. doi:10.1007/978-3-662-54580-5_5.
- 12 Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal logics for hyperproperties. In Martín Abadi and Steve Kremer, editors, *POST 2014*, volume 8414 of *LNCS*, pages 265–284. Springer, 2014. doi:10.1007/978-3-642-54792-8_15.
- 13 Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *J. Comput. Secur.*, 18(6):1157–1210, 2010. doi:10.3233/JCS-2009-0393.
- 14 Norine Coenen, Bernd Finkbeiner, Christopher Hahn, and Jana Hofmann. The hierarchy of hyperlogics. In *LICS 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785713.

- 15 Rina S. Cohen and Janusz A. Brzozowski. Dot-depth of star-free events. *J. Comput. Syst. Sci.*, 5(1):1–16, 1971. doi:10.1016/S0022-0000(71)80003-X.
- 16 Rayna Dimitrova, Bernd Finkbeiner, and Hazem Torfah. Probabilistic hyperproperties of Markov decision processes. In Dang Van Hung and Oleg Sokolsky, editors, *ATVA 2020*, volume 12302 of *LNCS*, pages 484–500. Springer, 2020. doi:10.1007/978-3-030-59152-6_27.
- 17 E. Allen Emerson and Joseph Y. Halpern. “Sometimes” and “not never” revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986. doi:10.1145/4904.4999.
- 18 Bernd Finkbeiner. Model checking algorithms for hyperproperties (invited paper). In Fritz Henglein, Sharon Shoham, and Yakir Vizel, editors, *VMCAI 2021*, volume 12597 of *LNCS*, pages 3–16. Springer, 2021. doi:10.1007/978-3-030-67067-2_1.
- 19 Bernd Finkbeiner and Christopher Hahn. Deciding hyperproperties. In Josée Desharnais and Radha Jagadeesan, editors, *CONCUR 2016*, volume 59 of *LIPICs*, pages 13:1–13:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.CONCUR.2016.13.
- 20 Bernd Finkbeiner, Christopher Hahn, and Tobias Hans. MGHyper: Checking satisfiability of HyperLTL formulas beyond the $\exists^*\forall^*$ fragment. In *ATVA 2018*, volume 11138 of *LNCS*, pages 521–527. Springer, 2018. doi:10.1007/978-3-030-01090-4_31.
- 21 Bernd Finkbeiner, Christopher Hahn, Jana Hofmann, and Leander Tentrup. Realizing omega-regular hyperproperties. In Shuvendu K. Lahiri and Chao Wang, editors, *CAV 2020, Part II*, volume 12225 of *LNCS*, pages 40–63. Springer, 2020. doi:10.1007/978-3-030-53291-8_4.
- 22 Bernd Finkbeiner, Christopher Hahn, Philip Lukert, Marvin Stenger, and Leander Tentrup. Synthesis from hyperproperties. *Acta Informatica*, 57(1-2):137–163, 2020. doi:10.1007/s00236-019-00358-2.
- 23 Bernd Finkbeiner, Christopher Hahn, and Marvin Stenger. EAHyper: Satisfiability, Implication, and Equivalence Checking of Hyperproperties. In Rupak Majumdar and Viktor Kuncak, editors, *CAV 2017, Part II*, volume 10427 of *LNCS*, pages 564–570. Springer, 2017. doi:10.1007/978-3-319-63390-9_29.
- 24 Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, and Leander Tentrup. RVHyper: A runtime verification tool for temporal hyperproperties. In Dirk Beyer and Marieke Huisman, editors, *TACAS 2018, Part II*, volume 10806 of *LNCS*, pages 194–200. Springer, 2018. doi:10.1007/978-3-319-89963-3_11.
- 25 Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. Algorithms for Model Checking HyperLTL and HyperCTL*. In Daniel Kroening and Corina S. Pasareanu, editors, *CAV 2015, Part I*, volume 9206 of *LNCS*, pages 30–48. Springer, 2015. doi:10.1007/978-3-319-21690-4_3.
- 26 Bernd Finkbeiner and Martin Zimmermann. The First-Order Logic of Hyperproperties. In *STACS 2017*, volume 66 of *LIPICs*, pages 30:1–30:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.STACS.2017.30.
- 27 Marie Fortin, Louwe B. Kuijjer, Patrick Totzke, and Martin Zimmermann. HyperLTL satisfiability is Σ_1^1 -complete, HyperCTL* satisfiability is Σ_1^2 -complete. *arXiv*, 2021. arXiv:2105.04176.
- 28 Jens Oliver Gutsfeld, Markus Müller-Olm, and Christoph Ohrem. Propositional dynamic logic for hyperproperties. In Igor Konnov and Laura Kovács, editors, *CONCUR 2020*, volume 171 of *LIPICs*, pages 50:1–50:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CONCUR.2020.50.
- 29 David Harel. Recurring Dominoes: Making the Highly Undecidable Highly Understandable. *North-Holland Mathematical Studies*, 102:51–71, 1985. doi:10.1016/S0304-0208(08)73075-5.
- 30 Peter G. Hinman. *Recursion-Theoretic Hierarchies*. Perspectives in Logic. Cambridge University Press, 2017. doi:10.1017/9781316717110.
- 31 Hsi-Ming Ho, Ruoyu Zhou, and Timothy M. Jones. Timed hyperproperties. *Information and Computation*, page 104639, 2020. doi:10.1016/j.ic.2020.104639.
- 32 Andreas Krebs, Arne Meier, Jonni Virtema, and Martin Zimmermann. Team semantics for the specification and verification of hyperproperties. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *MFCS 2018*, volume 117 of *LIPICs*, pages 10:1–10:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.MFCS.2018.10.

- 33 Corto Mascle and Martin Zimmermann. The keys to decidable HyperLTL satisfiability: Small models or very simple formulas. In Maribel Fernández and Anca Muscholl, editors, *CSL 2020*, volume 152 of *LIPICs*, pages 29:1–29:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CSL.2020.29.
- 34 Amir Pnueli. The temporal logic of programs. In *FOCS 1977*, pages 46–57. IEEE, October 1977. doi:10.1109/SFCS.1977.32.
- 35 Markus N. Rabe. *A temporal logic approach to information-flow control*. PhD thesis, Saarland University, 2016. URL: <http://scidok.sulb.uni-saarland.de/volltexte/2016/6387/>.
- 36 Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, MA, USA, 1987.
- 37 Wolfgang Thomas. A combinatorial approach to the theory of omega-automata. *Inf. Control.*, 48(3):261–283, 1981. doi:10.1016/S0019-9958(81)90663-X.
- 38 Wolfgang Thomas. Classifying regular events in symbolic logic. *J. Comput. Syst. Sci.*, 25(3):360–376, 1982. doi:10.1016/0022-0000(82)90016-2.
- 39 Jonni Virtema, Jana Hofmann, Bernd Finkbeiner, Juha Kontinen, and Fan Yang. Linear-time temporal logic with team semantics: Expressivity and complexity. *arXiv*, 2020. arXiv:2010.03311.

Matching Patterns with Variables Under Hamming Distance

Paweł Gawrychowski ✉ 

Faculty of Mathematics and Computer Science, University of Wrocław, Poland

Florin Manea ✉ 

Computer Science Department and Campus-Institut Data Science, Göttingen University, Germany

Stefan Siemer ✉ 

Computer Science Department, Göttingen University, Germany

Abstract

A pattern α is a string of variables and terminal letters. We say that α matches a word w , consisting only of terminal letters, if w can be obtained by replacing the variables of α by terminal words. The matching problem, i.e., deciding whether a given pattern matches a given word, was heavily investigated: it is NP-complete in general, but can be solved efficiently for classes of patterns with restricted structure. In this paper, we approach this problem in a generalized setting, by considering approximate pattern matching under Hamming distance. More precisely, we are interested in what is the minimum Hamming distance between w and any word u obtained by replacing the variables of α by terminal words. Firstly, we address the class of regular patterns (in which no variable occurs twice) and propose efficient algorithms for this problem, as well as matching conditional lower bounds. We show that the problem can still be solved efficiently if we allow repeated variables, but restrict the way the different variables can be interleaved according to a locality parameter. However, as soon as we allow a variable to occur more than once and its occurrences can be interleaved arbitrarily with those of other variables, even if none of them occurs more than once, the problem becomes intractable.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Formal languages and automata theory

Keywords and phrases Pattern with variables, Matching algorithms, Hamming distance, Conditional lower bounds, Patterns with structural restrictions

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.48

Related Version *Full Version*: <https://arxiv.org/abs/2106.06249> [30]

Funding The work of the two authors from Göttingen was supported by the DFG-grant 389613931.

1 Introduction

A *pattern* (with variables) is a string which consists of *terminal letters* (e.g., a, b, c), treated as constants, and *variables* (e.g., x_1, x_2). A pattern is mapped to a word by substituting the variables by strings of terminals. For example, $x_1x_1babx_2x_2$ can be mapped to **aaaababbb** by the substitution ($x_1 \rightarrow \mathbf{aa}, x_2 \rightarrow \mathbf{b}$). If a pattern α can be mapped to a string of terminals w , we say that α matches w . The problem of deciding whether there exists a substitution which maps a given pattern α to a given word w is called the *matching problem*.

Patterns with variables and their matching problem appear in various areas of theoretical computer science. In particular, the matching problem is a particular case of the satisfiability problem for word equations. These are equations whose both sides are patterns with variables and whose solutions are substitutions that map both sides to the same word [37]; in the pattern matching problem, one side of the input equation is a string of terminals. Patterns with variables occur also in combinatorics on words (e.g., unavoidable patterns [38]), stringology (e.g., generalized function matching [2]), language theory (e.g., pattern languages [3]), or



© Paweł Gawrychowski, Florin Manea, and Stefan Siemer;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 48; pp. 48:1–48:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

database theory (e.g., document spanners [27, 26, 19, 44]). In a more practical setting, patterns with variables are used in connection to extended regular expressions with backreferences [14, 29, 25, 28], used in various programming languages.

The *matching problem* is NP-complete [3] in general. This is especially unfortunate for some computational tasks on patterns which implicitly solve the matching problem and are thus intractable as well. For instance, in algorithmic learning theory, this is the case for the task of computing *descriptive patterns* for finite sets of words [3, 21]. Such descriptive patterns are useful for the inductive inference of pattern languages, a prominent example of a language class which can be inferred from positive data (see, the survey [46] and the references therein). This and many other applications of pattern matching provide a good motivation to identify cases in which the matching problem becomes tractable. A natural approach to this task is to consider restricted classes of patterns. A thorough analysis [42, 45, 23, 24, 22, 43] of the complexity of the matching problem has provided several subclasses of patterns for which the matching problem is in P, when some structural parameters of patterns are bounded by constants. Prominent examples in this direction are patterns with a bounded number of repeated variables occurring in a pattern, patterns with bounded scope coincidence degree [42], or patterns with bounded locality [18]. The formal definitions of these parameters are given in Section 4, and corresponding efficient matching algorithms be found in [22, 18], but, to give an intuition, we mention that they are all numerical parameters which describe the structure of patterns and parameterize the complexity of the matching algorithms. That is, in all cases, if the respective parameter equals k , the matching algorithm runs in $O(n^{ck})$ for some constant c , and, moreover, the matching problem can be shown to be $W[1]$ -hard w.r.t. the respective parameter. A more general approach [42] introduces the notion of treewidth of patterns, and shows that the matching problem can be solved in $O(n^{2k+4})$ time for patterns with bounded treewidth k . The algorithms resulting from this general theory are less efficient than the specialized ones, while the matching problem remains $W[1]$ -hard w.r.t. treewidth of patterns. See also the survey [39].

In this paper, we extend the study of patterns which can be matched efficiently to the case of approximate matching: we allow mismatches between the word w and the image of α under a substitution. More precisely, we consider two problems. In the decision problem MisMatch_P we are interested in deciding, for a given pattern α from a class P , a given word w , and a non-negative integer Δ whether there exists a variable-substitution h such that the word $h(\alpha)$ has at most Δ mismatches to the word w ; in other words, the Hamming distance $d_{\text{HAM}}(h(\alpha), w)$ between $h(\alpha)$ and w is at most Δ . Alternatively, we consider the corresponding minimisation problem MinMisMatch_P of computing $d_{\text{HAM}}(\alpha, w) = \min\{d_{\text{HAM}}(h(\alpha), w) \mid h \text{ is a substitution of the variables in } \alpha\}$.

As most real-world textual data (e.g., involving genetic data or text written by humans) contains errors, considering string-processing algorithms in an approximate setting is natural and has been heavily investigated. See, e.g., the recent papers [16, 32, 31, 47], and the references therein, as well as classical results such as [1, 41, 35]. Closer to the topic of this paper, the problem of approximate pattern matching was also considered in the context of regular expression matching – see [6, 41] and the references therein. Continuing this line of research, we initiate a study of approximate matching problems for patterns with variables. Intuitively, in our problems, we ask if the input word w is a few mismatches away from matching the pattern α , i.e., if w can be seen as a slightly erroneous version of a word which exactly matches α .

■ **Table 1** Our results are listed in columns 3 and 4. We assume $|w| = n$, $|\alpha| = m$, $|\text{var}(\alpha)| = p$.

Class	$\text{Match}(w, \alpha)$	$\text{MisMatch}(w, \alpha, \Delta)$	$\text{MinMisMatch}(w, \alpha)$
Reg	$O(n)$ [folklore]	$O(n\Delta)$ matching cond. lower bound	$O(nd_{\text{HAM}}(\alpha, w))$ matching cond. lower bound
1Var	$O(n)$ [folklore]	$O(n)$	$O(n)$
NonCross	$O(nm \log n)$ [22]	$O(n^3 p)$	$O(n^3 p)$
1RepVar $k = \# x$ -blocks	$O(n^2)$ [22]	$O(n^{k+2} m)$ W[1]-hard w.r.t. k	$O(n^{k+2} m)$, PTAS W[1]-hard w.r.t. k no FPTAS (if $FPT \neq W[1]$)
kLOC	$O(mkn^{2k+1})$ [18] W[1]-hard w.r.t. k	$O(n^{2k+2} m)$ W[1]-hard w.r.t. k	$O(n^{2k+2} m)$ W[1]-hard w.r.t. k no FPTAS (if $FPT \neq W[1]$)
kSCD	$O(m^2 n^{2k})$ [22] W[1]-hard w.r.t. k	NP-hard for $k \geq 2$	NP-hard for $k \geq 2$
kRepVar	$O(n^{2k})$ [22] W[1]-hard w.r.t. k	NP-hard for $k \geq 1$	NP-hard for $k \geq 1$
k -bounded treewidth	$O(n^{2k+4})$ [42] W[1]-hard w.r.t. k	NP-hard for $k \geq 3$	NP-hard for $k \geq 3$

Our Contribution. Our results are summarized in Table 1. In that table, we describe the results we obtained for the problems MisMatch_P and MinMisMatch_P (introduced informally above and formally in Section 2) for a series of classes P of patterns for which the matching problem Match can be solved in polynomial time. The classes P we consider are the following: The class **Reg** of regular patterns, which contain at most one occurrence of any variable; the class **1Var** of unary patterns, which contain several occurrences of a single variable and terminals; the class **NonCross** of non-cross-patterns, which can be factorized in multiple **1Var**-patterns whose variables are pairwise different; the class **1RepVar** of one-repeated-variables, where only one variable (say x) is allowed to occur more than once; the classes **kLOC** of k -local patterns and **kSCD** of patterns with scope coincidence degree at most k , defined formally in Section 4; the class **kRepVar** of k -repeated-variables, where only k variables are allowed to occur more than once. We also (indirectly) obtain a lower bound for the complexity of MisMatch and MinMisMatch in the case of patterns with treewidth at most k .

Interestingly, for **Reg** we obtain matching upper and conditional lower bounds. As regular patterns are, in fact, a particular case of regular expressions, it is worth mentioning that, due to the conditional lower bounds from [4] on exact regular expression matching, it is not to be expected that the general case of matching regular-expressions under Hamming distance can be solved as efficiently as the case of regular patterns. Regarding patterns with repeated variables, we note that while in the case when the number of repeated variables, the scope coincidence degree, or the treewidth was bounded by a constant, polynomial-time algorithms for the exact matching problem were obtained. This does not hold in our approximate setting, unless $P=NP$. Only the locality measure has the same behaviour as in the case of exact matching: $\text{MisMatch}_{\text{kLOC}}$ and $\text{MinMisMatch}_{\text{kLOC}}$ can still be solved in polynomial time for constant k . In the simpler case of **1RepVar**-patterns, the locality corresponds to the number of x -blocks, so, if this is bounded by a constant, the two problems we consider can be solved in polynomial time.

The paper is organized as follows: after some preliminaries, we present in detail the results on **Reg**-patterns. Then we overview the results on patterns with repeated variables.

Future Work. While our results paint a detailed image of the complexity of `MisMatch` and `MinMisMatch` for some prominent classes of patterns for which the matching problem can be solved efficiently, some continuations of this work can be easily identified. Following [22], it would be interesting to try to optimise the algorithms for all classes from the table (except `Reg`, where the upper and conditional lower bounds match). In the case of `Reg`, it would be interesting to consider the problem for regular patterns with a constant number of variables; already in the case of two variables (also known as approximate string matching under Hamming distance) the known complexity upper and lower bounds do not match anymore [31, 47]. Another direction is to consider the two problems for other distance functions (e.g., edit distance) instead of the Hamming distance. Finally, it would be interesting if the applications of pattern matching in the area of algorithmic learning theory can be formulated (and still remain interesting) in this approximate setting.

2 Preliminaries

Let Σ be a finite alphabet of *terminal letters*. Let Σ^* be the set of all words and ε the empty word. The concatenation of k words w_1, w_2, \dots, w_k is written $\prod_{i=1}^k w_i$. The set Σ^+ is defined as $\Sigma^* \setminus \{\varepsilon\}$. For $w \in \Sigma^*$ the length of w is defined the number of symbols of w , and denoted as $|w|$. Further, let $\Sigma^n = \{w \in \Sigma^* \mid |w| = n\}$ and $\Sigma^{\leq n} = \bigcup_{i=0}^n \Sigma^i$. The letter on position i of w , for $1 \leq i \leq |w|$, is denoted by $w[i]$. For $w \in \Sigma^+$ and $x, y, z \in \Sigma^*$, the word y is a factor of w , if $w = xyz$; moreover, if $x = \varepsilon$ (respectively, $z = \varepsilon$), then y is called a prefix (respectively, suffix) of w . Let $w[i : j] = w[i] \cdots w[j]$ be the factor of w starting on position i and ending on position j ; if $i > j$ then $w[i : j] = \varepsilon$. By $[i : j]$ we denote the set $\{i, i + 1, \dots, j\}$ and $D[i : j]$ denotes a subarray of D whose positions are indexed by the numbers in $[i : j]$.

Let $\mathcal{X} = \{x_1, x_2, x_3, \dots\}$ be a set of *variables*. For the set of terminals Σ and the set of variables \mathcal{X} with $\Sigma \cap \mathcal{X} = \emptyset$, a pattern α is a word containing both terminals and variables, i.e., an element of $PAT_\Sigma = (\mathcal{X} \cup \Sigma)^+$. The set of all patterns, over all terminal-alphabets, is denoted $PAT = \bigcup_\Sigma PAT_\Sigma$. Given a word or a pattern γ , for the smallest sets (w.r.t. inclusion) $B \subseteq \Sigma$ and $Y \subseteq \mathcal{X}$ with $\gamma \in (B \cup Y)^*$, define the set of terminal symbols in γ , denoted by $\text{alph}(\gamma) = B$, and the set of variables of γ , denoted by $\text{var}(\gamma) = Y$. For any symbol $t \in \Sigma \cup \mathcal{X}$ and $\alpha \in PAT_\Sigma$, $|\alpha|_t$ denotes the number of occurrences of t in α .

A substitution (on the variables of α) is a mapping $h : \text{var}(\alpha) \rightarrow \Sigma^*$. For every $x \in \text{var}(\alpha)$, we say that x is substituted by $h(x)$ and $h(\alpha)$ denotes the word obtained by substituting every occurrence of a variable x in α by $h(x)$ and leaving all the terminals unchanged. We say that the pattern α matches a word $w \in \Sigma^+$, if there exists a substitution $h : \text{var}(\alpha) \rightarrow \Sigma^*$ such that $h(\alpha) = w$. The Matching Problem is defined for any family of patterns $P \subseteq PAT$:

Exact Matching Problem for P : **Match_P**

Input: A pattern $\alpha \in P$, with $|\alpha| = m$, a word w , with $|w| = n$.

Question: Is there a substitution h with $h(\alpha) = w$?

In this paper, we will consider an extension of the Matching Problem, in which we allow mismatches between the image of the pattern under a substitution and the matched word.

For words $w_1, w_2 \in \Sigma^*$ with $|w_1| = |w_2|$, the Hamming distance between w_1 and w_2 is defined as $d_{\text{HAM}}(w_1, w_2) = |\{i \mid w_1[i] \neq w_2[i] \wedge 1 \leq i \leq |w_1|\}|$. The Hamming distance describes, therefore, the number of mismatches between two words. For a pattern α and a word w , we can define the Hamming distance between α and w as $d_{\text{HAM}}(\alpha, w) = \min\{d_{\text{HAM}}(h(\alpha), w) \mid h \text{ is a substitution of the variables of } \alpha\}$. With these definitions we can introduce two new pattern matching problems for families of patterns $P \subseteq PAT$. In the first problem, we allow

for a certain distance Δ between the image $h(\alpha)$ of α under a substitution h and the target word w instead of searching for an exact matching. In the second problem, we are interested in finding the substitution h such that the number of mismatches between $h(\alpha)$ and the target word w is minimal, over all possible choices of h .

Approximate Matching Decision Problem for P : **MisMatch $_P$**

Input: A pattern $\alpha \in P$, with $|\alpha| = m$, a word w , with $|w| = n$, an integer $\Delta \leq m$.

Question: Is $d_{\text{HAM}}(\alpha, w) \leq \Delta$?

Approximate Matching Minimisation Problem for P : **MinMisMatch $_P$**

Input: A pattern $\alpha \in P$, with $|\alpha| = m$, a word w , with $|w| = n$.

Question: Compute $d_{\text{HAM}}(\alpha, w)$.

When analysing the number of mismatches between $h(\alpha)$ and w we need to argue about the number of mismatches between corresponding factors of $h(\alpha)$ and w , i.e., the factors occurring between the same positions i and j in both words. To simplify the presentations, for a substitution h that maps a pattern α to a word of the same length as w , we will call the factors $h(\alpha)[i : j]$ and $w[i : j]$ aligned under h . We omit h when it is clear from the context. Moreover, saying that we align a factor $\alpha[i : j]$ to a factor $w[i' : j']$ with a minimal number of mismatches, we mean that we are looking for a substitution h such that $|h(\alpha)| = |w|$, $h(\alpha[i : j])$ is aligned to $w[i' : j']$ under h , and the resulting number of mismatches between $h(\alpha[i : j])$ and $w[i' : j']$ is minimal w.r.t. all other choices for the substitution h .

We make some preliminary remarks. Firstly, in all the problems we consider here, we can assume that the pattern α starts and ends with variables, i.e., $\alpha = x\alpha'y$, with α' pattern and x and y variables. Indeed, if this would not be the case, we could simply reduce the problems by considering them for inputs α' and the word w' obtained by removing from w the prefix and suffix aligned, respectively, to the maximal prefix of α which contains only terminals and the maximal suffix of α which contains only terminals. Clearly, in the case of the exact-matching problem the respective prefixes (suffixes) of w and α must match exactly, while in the case of the approximate-matching problems one needs to account for the mismatches created by these prefixes and suffixes. So, from now on, we will work under the assumption that the patterns we try to align to words start and end with variables.

Secondly, solving **Match $_P$** is equivalent to solving **MisMatch $_P$** for $\Delta = 0$. Also, in a general framework, **MinMisMatch $_P$** can be solved by combining the solution of the decision problem **MisMatch $_P$** with a binary search on the value of Δ . Given that the distance between α and w is at most $n = |w|$, one needs to use the solution for **MisMatch $_P$** a maximum of $\log n$ times in order to find the exact distance between α and w . Sometimes this can be done even more efficiently, as shown in Theorem 3.4. On the other hand, solving **MinMisMatch $_P$** leads directly to a solution for **MisMatch $_P$** .

3 Matching Regular Patterns with Mismatches

A pattern α is *regular* if $\alpha = w_0 \prod_{i=1}^M (x_i w_i)$, with $w_i \in \Sigma^*$. The class of regular patterns is denoted by **Reg**. For example, the pattern $\alpha_0 = \text{abxabyzbaab}$, with $\text{var}\alpha = \{x, y, z\}$ is in **Reg**.

In this section we consider **MisMatch $_{\text{Reg}}$** and **MinMisMatch $_{\text{Reg}}$** .

As mentioned already, a solution for **MisMatch $_{\text{Reg}}$** with distance $\Delta = 0$ is a solution to **Match $_{\text{Reg}}$** . The latter problem can be solved in $\mathcal{O}(n)$ by a greedy approach. As noted in Section 2, we can assume that $w_0 = w_M = \varepsilon$, so $\alpha = (\prod_{i=1}^{M-1} x_i w_i) x_M$. Thus, we identify the last occurrence $w[\ell + 1 : \ell + |w_{M-1}|]$ of w_{M-1} in w , assign the string $w[\ell + |w_{M-1}| + 1 : n]$ to x_M , and then recursively match the pattern $\alpha = (\prod_{i=1}^{M-2} x_i w_i) x_{M-1}$ to $w[1 : \ell]$.

In the following, we propose a solution for $\text{MinMisMatch}_{\text{Reg}}$ which generalizes this approach. Further, we will show a matching lower bound for any algorithm solving $\text{MinMisMatch}_{\text{Reg}}$.

An equivalent formulation of $\text{MinMisMatch}_{\text{Reg}}$ is to find factors $w[\ell_i + 1 : \ell_i + |w_i|]$, with $1 \leq i \leq M-1$, such that $\sum_{i=1}^{M-1} d_{\text{HAM}}(w_i, w[\ell_i + 1 : \ell_i + |w_i|])$ is minimal and $\ell_i + |w_i| + 1 \leq \ell_{i+1}$, for all $i \in \{1, \dots, M-2\}$. In other words, we want to find the $M-1$ factors $w[\ell_i + 1 : \ell_i + |w_i|]$, with i from 1 to $M-1$, such that these factors occur one after the other without overlapping in w , they correspond (in order, from left to right) to the words w_i , for i from 1 to $M-1$, and the total sum of mismatches between $w[\ell_i + 1 : \ell_i + |w_i|]$ and w_i , added up for i from 1 to $M-1$, is minimal.

To approach this problem we need the following data-structures-preliminaries.

Given a word w , of length n , we can construct in $O(n)$ -time *longest common suffix*-data structures which allow us to return in $O(1)$ -time the value $LCS_w(i, j) = \max\{|v| \mid v \text{ is a suffix of both } w[1 : i] \text{ and } w[1 : j]\}$. See [33, 34] and the references therein. Given a word w , of length n , and a word u , of length m , we can construct in $O(n+m)$ -time data structures which allow us to return in $O(1)$ -time the value $LCS_{w,u}(i, j) = \max\{|v| \mid v \text{ is a suffix of both } w[1 : i] \text{ and } u[1 : j]\}$. This is achieved by constructing LCS_w -data structures for wu , as above, and noting that $LCS_{w,u}(i, j) = \min(LCS_w(i, n+j), j)$.

The following two lemmas are based on the data structures defined above and the technique called kangaroo-jump [35]. Their respective proofs can be found in the Appendix B.

► **Lemma 3.1.** *Let w and u , with $|w| = |u| = n$, be two words and δ a non-negative integer. Assume that, in a preprocessing phase, we have constructed $LCS_{w,u}$ -data structures. We can compute $\min(\delta + 1, d_{\text{HAM}}(u, w))$ using $\delta + 1$ $LCS_{w,u}$ queries, so in $O(\delta)$ time.*

► **Lemma 3.2.** *Given a word w , with $|w| = n$, a word u , with $|u| = m < n$, and a non-negative integer δ , we can compute in $O(n\delta)$ time the array $D[m : n]$ with $n - m + 1$ elements, where $D[i] = \min(\delta + 1, d_{\text{HAM}}(w[i - m + 1 : i], u))$.*

The following result is the main technical tool of this section.

► **Theorem 3.3.** *$\text{MisMatch}_{\text{Reg}}$ can be solved in $O(n\Delta)$ time. For an accepted instance w, α, Δ of $\text{MisMatch}_{\text{Reg}}$ we also compute $d_{\text{HAM}}(\alpha, w)$ (which is upper bounded by Δ).*

Proof. Assume $\alpha = \prod_{i=1}^{M-1} (x_i w_i) x_M$ and let $\alpha_\ell = \prod_{i=\ell}^{M-1} (x_i w_i) x_M$, for $\ell \in \{1, \dots, M-1\}$.

A first observation is that the problem can be solved in a standard way by dynamic programming in $O(nm)$ time.

We only give the main idea behind this approach. We can compute the minimum number of mismatches $T[i][j]$ which can be obtained when aligning the suffix of length i of w to the suffix of length j of α , for all $i \leq n$ and $j \leq m$. Clearly, $T[i][j]$ can be computed based on the values $T[i+1][j+1]$ and, if $\alpha[j]$ is a variable, $T[i+1][j]$. The full technicalities of this standard approach are easy to obtain so we do not go into further details.

We present a more efficient approach below.

Our efficient algorithm starts with a preprocessing phase, in which we compute $LCS_{w,u}$ -data structures, where $u = \prod_{i=\ell}^{M-1} w_i$. This allows us to retrieve in constant time answers to LCS_{w,w_i} -queries, for $1 \leq i \leq M-1$.

In the main phase of our algorithm, we compute an $(M-1) \times \Delta$ matrix $\text{Suf}[\cdot][\cdot]$, where, for $\ell \leq M-1$ and $d \leq \Delta$, we have $\text{Suf}[\ell][d] = g$ if and only if $w[g..n]$ is the shortest suffix of w with $d_{\text{HAM}}(\alpha_\ell, w[g : n]) \leq d$.

Once more, we note that the elements of $\text{Suf}[\cdot][\cdot]$ can be computed by a relatively straightforward dynamic programming approach in $O(nM\Delta)$ time. But, the strategy we present here is more efficient than that.

In our algorithm, we first use Lemma 3.2 to compute $Suf[M-1][\cdot]$ in $O(n\Delta)$ time. We simply run the algorithm of that lemma on the input strings w and w_{M-1} and the integer Δ . We obtain an array $D[\cdot]$, where $D[i] = \min(\Delta + 1, d_{\text{HAM}}(w[i - |w_{M-1}| + 1 : i], w_{M-1}))$. We now go with j from $|w_{M-1}|$ to n and, if $D[j] \leq \Delta$, we set $Suf[M-1][D[j]] = j - |w_{M-1}| + 1$. It is clear that $h = Suf[M-1][d]$ will be the starting position of the shortest suffix $w[h : n]$ of w such that $d_{\text{HAM}}(w_{M-1}x_M, w[h : n]) \leq d$. Thus, $Suf[M-1][\cdot]$ was correctly computed, and the time needed to do so is $O(n\Delta)$.

Further, we describe how to compute $Suf[\ell][\cdot]$ efficiently, based on $Suf[\ell+1][\cdot]$ (for ℓ from $M-2$ down to 1). We use the following approach. We go through the positions i of w from right to left and maintain a queue Q . When i is considered, Q stores all elements d such that $Suf[\ell][d]$ was not computed yet until reaching that position, but $i < Suf[\ell+1][d]$. Accordingly, the fact that d is in Q means that with a suitable alignment of w_ℓ ending on position i , we could actually find an alignment with $\leq d$ mismatches of α_ℓ with $w[i - |w_\ell| + 1 : n]$: when Q contains $d, \dots, d - t$, for some $t \geq 0$, an alignment of w_ℓ to $w[i - |w_\ell| + 1 : i]$ with $\leq t$ mismatches would lead to an alignment of α_ℓ with $w[i - |w_\ell| + 1 : n]$ with $\leq d$ mismatches by extending the alignment of $\alpha_{\ell+1}$ to $w[Suf[\ell+1][d-t] : n]$. The values d present in Q at some point are ordered increasingly (the older values are larger), the array $Suf[\ell+1][\cdot]$ is also monotonically increasing, and, as $Suf[\ell][d]$ cannot be set before $Suf[\ell][d']$, for any d and d' such that $d' < d$, the queue Q is actually an interval of integers $[new : old]$, where new is the newest element of Q , and old the oldest one. When we consider position i of the word, if the alignment of w_ℓ ending on position i causes t mismatches, then to be able to set a value $Suf[\ell][d]$, with $d \in Q$, we need to have that $Suf[\ell+1][d-t] > i$. As $Suf[\ell+1][d] > Suf[\ell+1][d-t]$ and $d \in Q$, this means that $d-t \in Q$, so the number of mismatches t must be strictly upper bounded by $|Q|$, in order to be useful. Accordingly, when considering position i , we compute the number $t \leftarrow \min\{d_{\text{HAM}}(w_\ell, w[i - |w_\ell| + 1 : i]), |Q|\}$, and if $t < |Q|$ we set $Suf[\ell][d] \leftarrow i - |w_\ell| + 1$ for all d such that $d-t \in Q$; we also eliminate all these elements d from the queue. Before considering a new position i , we check if $i = Suf[\ell+1][new-1]$, and, if yes, we insert $new-1$ in Q and update $new \leftarrow new-1$.

This computation of $Suf[\ell][\cdot]$ is implemented in the following algorithm:

1. Initialization: We maintain a queue Q , which initially contains only the Δ .
Let $new \leftarrow \Delta$ (this is the top element of the queue).
2. Iteration: For $i = Suf[\ell+1][\Delta] - 1$ down to $|w_\ell|$ we execute the steps a, b, and c:
 - a. Using Lemma 3.1 we compute $t \leftarrow \min(d_{\text{HAM}}(w_\ell, w[i - |w_\ell| + 1 : i]), |Q|)$.
 - b. If $t < |Q|$, we remove from Q all elements d , such that $d-t \geq new$, and set, for each of them, $Suf[\ell][d] \leftarrow i - |w_\ell| + 1$.
 - c. If $Suf[\ell+1][top-1] = i$ then we insert $top-1$ in Q and $top \leftarrow top-1$. Else, if $Suf[\ell+1][top-1] = 0$ then set $i \leftarrow 0$ and exit the loop.
3. Filling-in the remaining positions: Set all the positions of $Suf[\ell][\cdot]$ which were not filled during the above while-loop to 0.

The matrix $Suf[\cdot][\cdot]$ is computed correctly by the above algorithm, as it can be shown by the following inductive argument.

To show that $Suf[\ell][\cdot]$ is computed correctly by our algorithm, under the assumption that $Suf[\ell+1][\cdot]$ was correctly computed, we make several observations.

Firstly, it is clear that $Suf[\ell+1][d] \leq Suf[\ell+1][d+1]$. Secondly, when computed correctly, $Suf[\ell][d]$ should be the rightmost position g of w such that $d_{\text{HAM}}(w[g : n], w_\ell) = t \leq d$ and $Suf[\ell+1][d-t] \geq g + |w_\ell|$. Clearly, if $Suf[\ell][d+1] \neq 0$, then $Suf[\ell][d] < Suf[\ell][d+1]$.

Regarding the algorithm described in the main part of the paper, it is important to observe that the queue Q is ordered increasingly (i.e., the newer is an element in Q , the smaller it is) and the elements of Q form an interval $[new : old]$.

Now, let us show the correctness of the algorithm.

Let d be a non-negative integer, $d \leq \Delta$. Assume that our algorithm sets $Suf[\ell][d] = g$, with $g > 0$.

This means that d was removed from the queue in step 2.b when the for-loop was executed for $i = g + |w_\ell| - 1$. The reason for this removal was that $d_{\text{HAM}}(w[g : g + |w_\ell| - 1], w_\ell) = t \leq |Q| - 1$. Hence, in this step we have removed exactly those elements δ such that $new \leq \delta - t$. Accordingly, we also have that $new \leq d - t$ holds. Let $g' = Suf[\ell + 1][new]$. We thus have $g' > i = g + |w_\ell| - 1$, $d_{\text{HAM}}(\alpha_{\ell+1}, w[g' : n]) \leq new$, and $d_{\text{HAM}}(w_\ell x_\ell, w[g : g' - 1]) = t$. Putting this all together, we get that $d_{\text{HAM}}(\alpha_\ell, w[g : n]) \leq new + t \leq d$.

Now, assume for the sake of a contradiction, that there exists $g'' > g$ such that $d_{\text{HAM}}(\alpha_\ell, w[g'' : n]) \leq d$, i.e., $w[g : n]$ is not the shortest suffix s of w such that $d_{\text{HAM}}(\alpha_\ell, s) \leq d$. In this case, there exists d'' such that $g'' + |w_\ell| - 1 < Suf[\ell + 1][d'']$ and $d'' + d_{\text{HAM}}(w[g'' : g'' + |w_\ell| - 1], w_\ell) \leq d$. Because d is in Q when $i = g + |w_\ell| - 1$ is reached in the for-loop, then d must also be in Q when $i'' = g'' + |w_\ell| - 1$ is reached in the for-loop, because $i < i'' < Suf[\ell + 1][d''] \leq Suf[\ell + 1][d]$. In fact, as $Suf[\ell + 1][d] \geq Suf[\ell + 1][d''] > i''$, it follows that d'' must also be in Q when i'' is reached. Thus, $g \geq d - d''$ and, as we have seen above, $d - d'' \geq d_{\text{HAM}}(w[g'' : g'' + |w_\ell| - 1], w_\ell)$. Moreover, if new'' is the element on the top of the queue when i'' is reached, we have that $new'' \leq d''$. Hence, $new'' + d_{\text{HAM}}(w[g'' : g'' + |w_\ell| - 1], w_\ell) \leq d'' + d_{\text{HAM}}(w[g'' : g'' + |w_\ell| - 1], w_\ell) \leq d$. Therefore, when i'' was reached, all the conditions needed to remove d from Q and set $Suf[\ell][d] \leftarrow g''$ were met. We have reached a contradiction with our assumption that $g'' > g$.

In conclusion, if our algorithm sets $Suf[\ell][d] = g$, with $g > 0$, then $w[g : n]$ is the shortest suffix of w such that $d_{\text{HAM}}(w[g : n], w_\ell) \leq d$. By an analogous argument as the one used above in our proof by contradiction, we can show that if our algorithm sets $Suf[\ell][d] = 0$ then there does not exist any suffix $w[g : n]$ of w such that $d_{\text{HAM}}(w[g : n], w_\ell) \leq d$.

This means that our algorithm computing $Suf[\cdot][\cdot]$ is correct.

To finalize the proof of the theorem, we note that, after computing the entire matrix $Suf[\cdot][\cdot]$, we can accept the instance w, α, Δ of $\text{MisMatch}_{\text{Reg}}$ if and only if there exists $d \leq \Delta$ such that $Suf[1][d] \neq 0$. Moreover, $d_{\text{HAM}}(\alpha, w) = \min(\{d \mid Suf[1][d] \neq 0\} \cup \{+\infty\})$.

In the following we show that this algorithm works in $O(n\Delta)$ time. We will compute the complexity of this algorithm using amortized analysis. Firstly, we observe that the complexity of the algorithm is proportional to the total number of LCS_{w, w_ℓ} -queries we compute in step 2.a, for each $\ell \leq M$ or, in other words, over all executions of the algorithm. Now, we observe that when position i of w is considered (for a certain ℓ), we do $|Q|$ many LCS_{w, w_ℓ} -queries. So, this means that we do one query per each current element of Q (and none if $|Q| = 0$). Thus, the number of queries corresponding to each pair (ℓ, d) which appears in Q at some point equals the number of positions considered between the step when it was inserted in Q and the step when it was removed from Q . This means $O(Suf[\ell + 1][d] - Suf[\ell][d])$ queries corresponding to (ℓ, d) . Summing this up for a fixed d and ℓ from 1 to $M - 2$ we obtain that the overall number of queries corresponding to a fixed δ is $O(Suf[M - 1][d]) = O(n)$. Adding this up for all $d \leq \Delta$, we obtain that the number of LCS -queries performed in our algorithm is $O(n\Delta)$. So, together with the complexity of the initialization of $Suf[M - 1][\cdot]$, the complexity of this algorithm is $O(n\Delta)$.

This algorithm outperforms the other two algorithms solving $\text{MinMisMatch}_{\text{Reg}}$ which we mentioned, and, for $\Delta = 0$, it is a reformulation of the greedy algorithm solving $\text{Match}_{\text{Reg}}$. ◀

► **Theorem 3.4.** $\text{MinMisMatch}_{\text{Reg}}$ can be solved in $O(n\Phi)$ time, where $\Phi = d_{\text{HAM}}(\alpha, w)$.

Proof. We use the algorithm of Theorem 3.3 for $\Delta = 2^i$, for increasing values of i starting with 1 and repeating until the algorithm returns a positive answer and computes $\Phi = d_{\text{HAM}}(\alpha, w)$. The algorithm is clearly correct. Moreover, the value of i which was considered last is such that $2^{i-1} < \Phi \leq 2^i$. So $i = \lceil \log_2 \Phi \rceil$, and the total complexity of our algorithm is $O(n \sum_{i=1}^{\lceil \log_2 \Phi \rceil} 2^i) = O(n\Phi)$. ◀

In order to show that $\text{MinMismatch}_{\text{Reg}}$ and $\text{Mismatch}_{\text{Reg}}$ cannot be solved by algorithms running polynomially faster than the algorithms from Theorems 3.3 and 3.4, we will reduce the Orthogonal Vectors problem OV [10] to $\text{Mismatch}_{\text{Reg}}$. The overall structure of our reduction is similar to the one used for establishing hardness of computing edit distance [5, 11] or LCS [12], however we needed to construct gadgets specific to our problem. We recall the OV problem.

Orthogonal Vectors: OV

Input: Two sets U, V consisting each of n vectors from $\{0, 1\}^d$, where $d \in \omega(\log n)$.

Question: Do vectors $u \in U, v \in V$ exist, such that u and v are orthogonal, i.e., for all $1 \leq k \leq d$, $v[k]u[k] = 0$ holds?

In general, for a vector $u = (u[1], \dots, u[d]) \in \{0, 1\}^d$, the bits $u[i]$ are called coordinates. It is clear that, for input sets U and V as in the above definition, one can solve OV trivially in $\mathcal{O}(n^2d)$ time. The following conditional lower bound is known for OV .

► **Lemma 3.5 (OV-Conjecture).** *OV can not be solved in $\mathcal{O}(n^{2-\epsilon}d^c)$ for any $\epsilon > 0$ and constant c , unless the Strong Exponential Time Hypothesis (SETH) fails.*

See [10, 48] and the references therein for a detailed discussion regarding conditional lower bounds related to OV . In this context, we can show the following result.

► **Theorem 3.6.** *$\text{Mismatch}_{\text{Reg}}$ can not be solved in $\mathcal{O}(|w|^h \Delta^g)$ time (or in $\mathcal{O}(|w|^h |\alpha|^g)$ time) with $h + g = 2 - \epsilon$ for some $\epsilon > 0$, unless the OV -Conjecture fails.*

Proof. We reduce OV to $\text{MinMismatch}_{\text{Reg}}$. For this, we consider an instance of OV : $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_n\}$, with $U, V \subset \{0, 1\}^d$. We transform this OV -instance into a $\text{Mismatch}_{\text{Reg}}$ -instance (α, w, Δ) , where $\Delta = n(d+1) - 1$. More precisely, we ensure that for the respective $\text{Mismatch}_{\text{Reg}}$ -instance, there exists a way to replace the variables with strings leading to exactly $n(d+1)$ mismatches between the image of α and w if and only if no two vectors u_i and v_j are orthogonal. But, if there exists at least one orthogonal pair of vectors u_i and v_j , there also exists a way to replace the variables of α such that the resulting string has strictly less than $n(d+1)$ mismatches to w . Both $|w|$ and $|\alpha|$ are in $\mathcal{O}(nd)$, and can be built in $\mathcal{O}(nd)$ time. The reduction consists of three main steps. First we will present a gadget for encoding the single coordinates of vectors u_i and v_i from U and V , respectively. Then we will show another gadget to encode a full vector of each respective set. And, finally, we will show how to assemble these gadgets of the vectors from set U into the word w and from V into α .

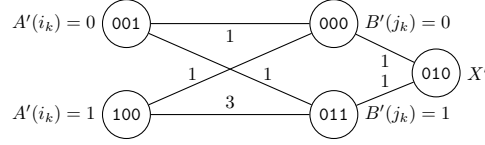
First gadget. Let $u_i = (u_i[1], u_i[2], \dots, u_i[d]) \in U, v_j = (v_j[1], v_j[2], \dots, v_j[d]) \in V$ and let k be a position of these vectors. We define the following gadgets:

$$A'(i_k) = \begin{cases} 001, & \text{if } u_i[k] = 0. \\ 100, & \text{if } u_i[k] = 1. \end{cases} \quad B'(j_k) = \begin{cases} 000, & \text{if } v_j[k] = 0. \\ 011, & \text{if } v_j[k] = 1. \end{cases}$$

Note that, when aligned, the pair of strings $(A'(i_k), B'(j_k))$ produces exactly one mismatch if and only if $u_i[k] \cdot v_j[k] = 0$; otherwise it produces three mismatches. So, $A'(i_k)$ and $B'(j_k)$ encode the single coordinates of u_i and v_j respectively.

48:10 Matching Patterns with Variables Under Hamming Distance

Further, we construct a gadget $X' = 010$ that produces always one mismatch if aligned to any of the strings $B'(j_k)$ corresponding to coordinates $v_j[k]$. See also Figure 1.



■ **Figure 1** Gadgets for the encoding of single coordinates of the vectors. On each edge we wrote the number of mismatches between the strings in the nodes connected by that edge.

Second gadget. The gadget $A(i)$ encodes the vector u_i , for $1 \leq i \leq n$, while the gadget $B(j)$ encodes the vector v_j , for $1 \leq j \leq n$. We construct these gadgets such that aligning $B(j)$ to $A(i)$ with a minimum number of mismatches yields exactly d mismatches, if the two corresponding vectors are orthogonal, and exactly $d + 1$ mismatches, otherwise. Moreover, we show that any other alignment of the gadgets $B(j)$ with other factors of w yields more mismatches.

In order to assemble the gadgets $A(i)$ and $B(j)$, for $1 \leq i, j \leq n$, we extend the terminal alphabet by three new symbols $\{\mathbf{a}, \mathbf{b}, \#\}$, as well as use two fresh variables x_j, y_j for each vector v_j . The gadgets $A(i)$, for all i , and, respectively, the gadgets $B(j)$, for all j , consist of the concatenation of the coordinate gadgets $A'(i_k)$ and, respectively, $B'(j_k)$ from left to right, in ascending order of k . Each two such consecutive gadgets $A'(i_k)$ and $A'(i_{k+1})$ (respectively, $B'(j_k)$ and $B'(j_{k+1})$) are separated by $\#\#\#$. We prepend to $A(i)$ the string \mathbf{bba} and append the string $\mathbf{bbb}X$, where $X = (X'\#\#\#)^{d-1}X'$. In the case of $B(j)$, we prepend $x_j\mathbf{bba}$ and append y_j . The full gadgets $A(i)$ and $B(j)$ are defined as follows.

- $A(i) = \mathbf{bba}A'(i_1)\#\#\#A'(i_2)\#\#\# \dots A'(i_d)\mathbf{bbb}X$
- $B(j) = x_j\mathbf{bba}B'(j_1)\#\#\#B'(j_2)\#\#\# \dots B'(j_d)y_j$.

For simplicity of the exposure, let $B'(j) = \mathbf{bba}B'(j_1)\#\#\#B'(j_2)\#\#\# \dots \#\#\#B'(j_d)$.

Note that $|A(i)|$ is the same for all i , so we can define $M = |A(i)|$.

Final assemblage. To define the word w , we use a new terminal $\$$. The word w is:

- $w = \$^M A(1)\$^M A(2)\$^M \dots A(n)\$^M A(1)\$^M A(2) \dots \$^M A(n)\M

To define α , we use two new fresh variables x and y . The pattern α is:

- $\alpha = x\$^M B(1)\$^M B(2)\$^M \dots \$^M B(n)\$^M y$.

The correctness of the reduction. We show that there exists a way to align α with w with $< n(d + 1)$ mismatches if and only if a pair of orthogonal vectors $u_i \in U$ and $v_j \in V$ exists. Otherwise, there exists an alignment of α to w with exactly $n(d + 1)$ mismatches.

To formally prove that the reduction fulfills this requirement, we proceed as follows.

A general idea: the repetition of the gadgets $A(i)$ in the word w guarantees that, if needed, a pair of gadgets $A(i)$ and $B(j)$, corresponding to the vectors $u_i \in U$ and, respectively, $v_j \in V$, can be aligned. More precisely, we can align $B'(j)$ to $\mathbf{bba}A'(i_1)\#\#\# \dots A'(i_d)$. The variables x, y and x_j, y_j , for $j \in \{1, \dots, n\}$, act as spacers: they allow us to align a string $B'(j)$ to the desired factor of w . This kind of alignment is enough for our purposes, as we only need to find one orthogonal pair of vectors, not all of them; however, we need enough space in w for the factors of α occurring before and after $B'(j)$, thus the repetition of the $A(i)$ gadgets.

We now analyse how a factor $B'(j)$ can be aligned to a factor of w . The main idea is to show that if there are no orthogonal vectors, then any alignment of $B'(j)$ to a factor of w creates at least $d + 1$ mismatches. Otherwise, we can align it with d mismatches only.

Case 1: $B'(j)$ is aligned to a factor $w[i : h]$ of w which starts with \$. Then the prefix **ba** of $B'(j)$ causes at least two mismatches, as the first **b** in **ba** is aligned to a \$ letter, while the **a** is aligned to either a **b** letter (from a **ba** factor) or a \$ letter. The rest of $B'(j)$ causes, overall, at least d mismatches, one per each group $B'(j_k)$. So, in this case, we have at least $d + 2$ mismatches caused by $B'(j)$.

Case 2: $B'(j)$ is aligned a factor $w[i : h]$ of w which ends with \$. Then, its prefix **ba** cannot be aligned to a factor **ba** of w . So, the **a** of the prefix **ba** of $B'(j)$ produces one mismatch, while the suffix $B'(j_d)$ causes at least 2 mismatches. The rest of $B'(j)$ causes at least $d - 1$ mismatches, one per each remaining group $B'(j_k)$. So, in this case, we have again at least $d + 2$ mismatches caused by $B'(j)$.

Case 3: $B'(j)$ is aligned exactly to the factor **ba** $A'(i_1)$ ###... $A'(i_d)$ and u_i and v_j are orthogonal, then $B'(j)$ causes exactly d mismatches.

Case 4: $B'(j)$ is aligned exactly to the factor **ba** $A'(i_1)$ ###... $A'(i_d)$ and u_i and v_j are not orthogonal, then $B'(j)$ causes at least $d + 2$ mismatches.

Case 5: $B'(j)$ is aligned exactly to the factor **bb** X , then $B'(j)$ causes $d + 1$ mismatches.

Case 6: $B'(j)$ is aligned to a factor starting strictly inside **ba** $A'(i_1)$ ###... $A'(i_d)$, then the prefix **ba** of $B'(j)$ cannot be aligned to a factor **ba** of w , so it causes at least two mismatches (from the alignment of **ba**). The rest of $B'(j)$ causes at least d mismatches, one per each group $B'(j_k)$. So, overall, $B'(j)$ causes at least $d + 2$ mismatches in this case.

To ease the understanding, cases 3 and 4 are illustrated in the following table: when aligning $A(i)$ to $B(j)$, to obtain the desired number of mismatches, we can match the parts of $A(i)$ to the parts of $B(j)$ as described in this table in the two cases 3. and 4.

Gadget	I	II	III	IV	mismatches
$A(i) =$	ε	ba $A'(i_1)$ ###...### $A'(i_d)$	bb X' ###...### X'	ε	
3. $B(j) =$	x_j	ba $B'(j_1)$ ###...### $B'(j_d)$	y_j	ε	d (in II)
4. $B(j) =$	ε	x_j	ba $B'(j_1)$ ###...### $B'(j_d)$	y_j	$d + 1$ (in IV)

Wrapping up, there are no other ways than those described in cases 1-6 above in which $B'(j)$ can be aligned to a factor of w . In particular, in order to reach an alignment with at most $n(d + 1) - 1$ mismatches, at least one $B'(j)$ should be aligned to a factor of w such that it only causes d mismatches (as in case 3). Thus, in that case we would have a pair of orthogonal vectors. Conversely, if there exist u_i and v_j which are orthogonal and $i \geq j$, then we can align $B'(j)$ to the occurrence of **ba** $A'(i_1)$ ###... $A'(i_d)$ from the first $A(i)$ and all the other gadgets $B'(\ell)$ to factors **bb** X , and obtain a number of $n(d + 1) - 1$ mismatches. Note that such an alignment is possible as there exist at least $j - 1$ factors **bb** X before the first $A(i)$ and at least n more occurrences of **bb** X after it; moreover the variables x_ℓ and y_ℓ can be used to align as desired the strings $B'(v_\ell)$ to the respective **bb** X factors of w . If there exist u_i and v_j which are orthogonal and $i < j$, then we can align $B'(j)$ to the occurrence of **ba** $A'(i_1)$ ### $A'(i_2)$ ###... $A'(i_d)$ from the second $A(i)$ and all the other gadgets $B'(\ell)$ to factors **bb** X , and obtain again a number of $n(d + 1) - 1$ mismatches. This is possible for similar reasons to the ones described above.

This shows that our reduction is correct. The instance of **OV** defined by U and V contains two orthogonal vectors if and only the instance of **MisMatch_{Reg}** defined by w, α , and $\Delta = n(d + 1) - 1$ can be answered positively. Moreover, the instance of **MisMatch_{Reg}** can be constructed in $O(nd)$ time and we have that $|w|, |\alpha|, \Delta \in \Theta(nd)$.

Assume now that there exists a solution of $\text{MisMatch}_{\text{Reg}}$ running in $O(|w|^g|\alpha|^h)$ with $g+h=2-\epsilon$ for some $\epsilon < 0$. This would lead to a solution for OV running in $O(nd+(nd)^{2-\epsilon})$, a contradiction to the OV -conjecture. Similarly, if there exists a solution of $\text{MisMatch}_{\text{Reg}}$ running in $O(|w|^g\Delta^h)$ with $g+h=2-\epsilon$ for some $\epsilon < 0$, then there exists a solution for OV running in $O(nd+(nd)^{2-\epsilon})$, a contradiction to the OV -conjecture. This proves our statement. \blacktriangleleft

► **Remark 3.7.** An immediate consequence of the previous theorem is that $\text{MinMisMatch}_{\text{Reg}}$ can not be solved in $\mathcal{O}(n^h d_{\text{HAM}}(\alpha, w)^g)$ time (or in $\mathcal{O}(|w|^h|\alpha|^g)$ time) with $h+g=2-\epsilon$ for some $\epsilon > 0$, unless the OV -Conjecture fails. Thus, as $d_{\text{HAM}}(\alpha, w) \leq |\alpha|$, $\text{MinMisMatch}_{\text{Reg}}$ and $\text{MisMatch}_{\text{Reg}}$ cannot be solved polynomially faster than our algorithms, unless the OV -Conjecture fails.

4 Patterns with Repeated Variables

In Section 3 we have shown that if no variable occurs more than once in the input pattern α , then the problems MisMatch and MinMisMatch can be solved in polynomial time. Let us now consider patterns where variables are allowed to occur more than once, i.e., patterns with repeated variables. Firstly, we recall two measures of the structural complexity of patterns.

For every variable $x \in \text{var}(\alpha)$, the scope of x in α is defined by $\text{sc}_\alpha(x) = [i : j]$, where i is the leftmost and j the rightmost occurrence of x in α . The scopes of the variables $x_1, \dots, x_k \in \text{var}(\alpha)$ coincide in α if $\bigcap_{i=1}^k \text{sc}(x_i) \neq \emptyset$. By $\text{scd}(\alpha)$ we denote the scope coincidence degree of α : the maximum number of variables in α whose scopes coincide. By kSCD we denote the class of patterns whose scope coincidence degree is at most k .

Given a pattern α , with p variables, a marking sequence of α is an ordering $x_1 < x_2 < \dots < x_p$ of $\text{var}(\alpha)$. The skeleton α_{var} of α is obtained from α by removing all the terminals. A marking of α_{var} w.r.t. a marking sequence $x_1 < x_2 < \dots < x_p$ of α is a p -steps procedure: in step i we mark all occurrences of variable x_i . The pattern α is called k -local if and only if there exists a marking sequence of $x_1 < x_2 < \dots < x_p$ of α such that, for i from 1 to p , the variables marked in the first i steps of the marking of α_{var} w.r.t. this marking sequence form at most k non-overlapping length-maximal factors in α_{var} ; the respective marking sequence is called witness for the k -locality of α . By kLOC we denote the class of k -local patterns. See [18, 15] for an extended discussion and examples regarding k -locality.

Several more particular classes which we consider in this context are the following:

- The class of unary patterns 1Var : $\alpha \in \text{1Var}$ if there exists $x \in X$ such that $\text{var}(\alpha) = \{x\}$; example: $\alpha_1 = \text{abxabxxbaab} \in \text{1Var}$.
- The class of one-repeated-variable patterns 1RepVar : $\alpha \in \text{1RepVar}$ if there exists at most one variable $x \in X$ such that $|\alpha|_x > 1$; example: $\alpha_2 = \text{abxyabzxxbaabv} \in \text{1RepVar}$.
- The class $\text{NonCross} = \text{1SCD}$, called the class of non-cross patterns; as examples, consider $\alpha_3 = \text{abxxyabzzbbvvvabvu} \in \text{NonCross} \setminus \text{1RepVar}$ and $\alpha_4 = \text{abxyabzxxbbvabx} \in \text{1RepVar} \setminus \text{NonCross}$. Note that $\alpha \in \text{NonCross}$ if and only if α can be written as the concatenation of several 1Var -patterns, whose variables are pairwise distinct. Thus, NonCross -patterns are 1-local.

Note that in a NonCross -pattern α , for any two variables $x, y \in \text{var}(\alpha)$, where the last occurrence of y is to the right of the first occurrence of x in α , we can actually write $\alpha = \beta x \gamma y \delta$ such that $x, y \notin \text{var}(\gamma)$, $x \notin \text{var}(\delta)$, and $y \notin \text{var}(\beta)$. In other words, there are no interleaved occurrences of two variables. Moreover, if $\alpha \in \text{NonCross}$, then α is 1-local: the marking sequence is obtained by ordering the variables according to the position of their first

occurrence. Clearly, $1\text{Var} \subset 1\text{RepVar}$ and $1\text{Var} \subset \text{NonCross}$, but 1RepVar and NonCross are incomparable. Indeed, if $\alpha \in \text{NonCross}$ then α is 1-local and 1RepVar contains patterns α with $\text{scd}(\alpha) = 2$.

Further, if α is a pattern and $x \in \text{var}(\alpha)$, then an x -block is a factor $\alpha[i : j]$ such that $\alpha[i : j] \in 1\text{Var}$ with $\text{var}(\alpha[i : j]) = x$ and it is length-maximal with this property: it cannot be extended to the right or to the left without introducing a variable different from x .

The next lemma is fundamental for the results of this section.

► **Lemma 4.1.** *Given a set of words $w_1, \dots, w_p \in \Sigma^m$, we can find in $O(|\Sigma| + mp)$ a median string for $\{w_1, \dots, w_p\}$, i.e. a string w such that $\sum_{j=1}^p d_{\text{HAM}}(w_i, w)$ is minimal.*

Proof. We will use an array C with Σ elements, called counters, indexed by the letters of Σ , and all initially set to 0. For each i between 1 and m , we count how many times each letter of Σ occurs in the multi-set $\{w_1[i], w_2[i], \dots, w_p[i]\}$ using C . Let $w[i]$ be the most frequent letter of this multi-set. After computing $w[i]$, we reset the counters which were changed in this iteration, and repeat the algorithm for $i + 1$. After going through all values of i , we return the word $w = w[1]w[2] \dots w[m]$ as the answer to the problem. The correctness of the algorithm is immediate, while its complexity is clearly $O(|\Sigma| + mp)$. ◀

The typical use of this lemma is the following: we identify the factors of w to which a repeated variable is aligned, and then compute the optimal assignment of this variable. Based on this, the following theorem can now be shown. The corresponding proof can be found in the full version of this paper [30].

► **Theorem 4.2.** $\text{MinMismatch}_{1\text{Var}}$ and $\text{Mismatch}_{1\text{Var}}$ can be solved in $O(n)$ time.

By a standard dynamic programming approach, we use the previous result to obtain a polynomial-time solution for $\text{MinMismatch}_{\text{NonCross}}$ based on the solution for $\text{MinMismatch}_{1\text{Var}}$ (in the statement, $p = |\text{var}(\alpha)|$). The corresponding proof can be found in the full version of this paper [30].

► **Theorem 4.3.** $\text{MinMismatch}_{\text{NonCross}}$ and $\text{Mismatch}_{\text{NonCross}}$ can be solved in $O(n^3p)$ time.

The results presented so far show that MinMismatch_P and Mismatch_P can be solved in polynomial time, as long as we do not allow interleaved occurrences of variables in the patterns of the class P . We now consider the case of 1RepVar -patterns, the simplest class of patterns which permits interleaved occurrences of variables. For simplicity, in the results regarding 1RepVar we assume that the variable which occurs more than once in the input pattern is denoted by x . The corresponding proof and the proof of the following more general result can be found in the Appendix B.

► **Theorem 4.4.** $\text{MinMismatch}_{1\text{RepVar}}$ and $\text{Mismatch}_{1\text{RepVar}}$ can be solved in $O(n^{k+2}m)$ time, where k is the number of x -blocks in the input pattern α .

► **Theorem 4.5.** $\text{MinMismatch}_{\text{kLOC}}$ and $\text{Mismatch}_{\text{kLOC}}$ can be solved in $O(n^{2k+2}m)$ time.

Note that NonCross -patterns are 1-local, while the locality of an 1RepVar -pattern is upper bounded by the number of x -blocks. However, the algorithms we obtained in those particular cases are more efficient than the ones which follow from Theorem 4.5.

The fact that Lemma 4.1 is used as the main building block for our results regarding Mismatch_P and MinMismatch_P for $P \in \{1\text{RepVar}, \text{kLOC}\}$, suggests that these problems could be closely related to the following well-studied problem [36, 20, 7, 13].

Consensus Patterns: CP

Input: k strings $w_1, \dots, w_k \in \Sigma^\ell$, integer $m \in \mathbb{N}$ with $m \leq \ell$, an integer $\Delta \leq mk$.

Question: Do the strings s , of length m , and s_1, \dots, s_k , factors of length m of each w_1, \dots, w_k , respectively, exist, such that $\sum_{i=1}^k d_{\text{HAM}}(s_i, s) \leq \Delta$?

Exploiting this connection, and following the ideas of [36], we can show the following theorem. In this theorem we restrict to the case when the input word w of $\text{MinMisMatch}_{1\text{RepVar}}$ is over $\Sigma = \{1, \dots, \sigma\}$ of constant size σ .

► **Theorem 4.6.** *For each constant $r \geq 3$, there exists an algorithm with run-time $O(n^{r+3})$ for $\text{MinMisMatch}_{1\text{RepVar}}$ whose output distance is at most $\min \left\{ 2, \left(1 + \frac{4\sigma-4}{\sqrt{\epsilon}(\sqrt{4r+1}-3)} \right) \right\} d_{\text{HAM}}(\alpha, w)$.*

The proof can be found in the Appendix B. It remains open whether other algorithmic results related to CP (such as those from, e.g., [8, 9, 40]) apply to our setting too.

In the following we show two hardness results which explain why the algorithms in Theorems 4.4 and 4.6 are interesting.

► **Theorem 4.7.** *$\text{MisMatch}_{1\text{RepVar}}$ is $W[1]$ -hard w.r.t. the number of x -blocks.*

Proof. We reduce CP to $\text{MisMatch}_{1\text{RepVar}}$, such that an instance of CP with k different input strings is mapped to an instance of $\text{MisMatch}_{1\text{RepVar}}$ with $k+1$ x -blocks (where x is the repeated variable), each containing exactly one occurrence of x .

Hence, we consider an instance of CP which consists of k strings $w_1, \dots, w_k \in \Sigma^\ell$ of length ℓ and two integer m, Δ defining the length of the target factors and the number of allowed mismatches, respectively.

The instance of $\text{MisMatch}_{1\text{RepVar}}$ which we construct consists of a text w and a pattern α , such that α contains $k+1$ x -blocks, each with exactly one occurrence of x , and is of polynomial size w.r.t. the size of the CP-instance. Moreover, the number of mismatches allowed in this instance of $\text{MisMatch}_{1\text{RepVar}}$ is $\Delta' = m + \Delta$. That is, if there exists a solution for the CP-instance with Δ allowed mismatches, then, and only then, we should be able to find a solution of the $\text{MisMatch}_{1\text{RepVar}}$ -instance with $\Delta + m$ mismatches.

The construction of the $\text{MinMisMatch}_{1\text{RepVar}}$ is realized in such a way that the word w encodes the input strings, while α creates the mechanism for selecting the string s and corresponding factors s_1, \dots, s_k . The general idea is that x should be mapped to s , and the factors to which the occurrences of x are aligned should correspond to the strings s_1, \dots, s_k .

The structure of the word w and that of the pattern α ensure that, in an alignment of α with w which cannot be traced back to a admissible solution for the CP-instance (that is, the occurrences of x are not aligned to factors of length m of the words w_1, \dots, w_k or x is not mapped to a string of length m) we have at least $M \gg \Delta'$ mismatches, hence it cannot lead to a positive answer for the constructed instance of $\text{MisMatch}_{1\text{RepVar}}$.

The reduction consists of three main steps. Firstly, we present a pair of gadgets to encode the relation of the strings w_i and their factors s_i , for i from 1 to k . Then, we present a second pair of gadgets, which ensures that, in a positive solution of $\text{MisMatch}_{1\text{RepVar}}$, the variable x can only be mapped to a string of length m , corresponding to the string s . Finally, we show how to assemble these gadgets into the input word w and the input pattern α for $\text{MisMatch}_{1\text{RepVar}}$.

First pair of gadgets. We introduce the new letters $\{a, b\}$, not contained in the input alphabet of the CP-instance, as well as the variable x and two fresh variables y_i, z_i , for each i from 1 to k . We construct the following two gadgets for each input string w_i with $1 \leq i \leq k$.

- A gadget to be included in w : $g_i = w_i \overbrace{a^M b^M \dots a^M b^M}^M$.
- A gadget to be included in α : $f_i = y_i x z_i \overbrace{a^M b^M \dots a^M b^M}^M$.

These gadgets allows us to align the i^{th} occurrence of x to an arbitrary factor of the word w_i , for i from 1 to k .

Second pair of gadgets. In this case, we use three new letters $\{c, d, \$\}$ which are not contained in the input alphabet of CP. Also, let $M = (k\ell)^2$. We define two new gadgets.

- A gadget to be included in w : $A_w = \overbrace{c^M d^M \dots c^M d^M}^M \m .
- A gadget to be included in α : $A_\alpha = \overbrace{c^M d^M \dots c^M d^M}^M x$.

These gadgets enforce that, in an alignment of α and w , the variable x is mapped to a string of length m , at the cost of exactly m extra mismatches. Note that, because $\Delta \leq km$, we have that $M \gg \Delta$.

Final assemblage. The word w and the pattern α are defined as follows.

- $w = g_1 g_2 \dots g_k A_w$ and $\alpha = f_1 f_2 \dots f_k A_\alpha$.

To wrap up, the instance of $\text{MinMismatch}_{1\text{RepVar}}$ is defined by $w, \alpha, \Delta + m$.

The correctness of the reduction. We will show that our reduction is correct by a detailed case analysis. We consider an alignment of α and w with minimal number of mismatches, and we make the following observations.

- A. Firstly, if every g_i is aligned to f_i , for i from 1 to k , it is immediate that x is mapped to a string of length m , as the last occurrence of x will be aligned to the $\m suffix of w . Thus, the total number of mismatches between α and w in an alignment with a minimum number of mismatches is upper-bounded by $(k+1)m$.
- B. Secondly, we assume, for the sake of a contradiction, that the length of the image of x is not m . If $|x| > m$ (respectively, $|x| < m$) then the prefix $(c^M d^M)^M$ of A_α is aligned to a factor of w which starts strictly to the left of (respectively, to the right of) the first position of the prefix $(c^M d^M)^M$ of A_w . It is not hard to see that this causes at least M mismatches. Indeed, in the case when $|x| > m$, if the factor $(c^M d^M)^M$ of α is aligned to a factor that starts at least M position to the left of the factor $(c^M d^M)^M$ of w , the conclusion is immediate; if the factor $(c^M d^M)^M$ starts less than M positions to the left of the factor $(c^M d^M)^M$ of w , then each group c^M in α will be aligned to a factor of w that includes at least a d letter, so we again reach the conclusion. In the case when $|x| < m$, then, again, each group c^M in α will be aligned to a factor of w that includes at least a d letter, so the alignment leads to at least M mismatches.

So, we can assume from now on that x is mapped to a string of length m . This also implies that A_α and A_w are aligned, so we will largely neglect them from now on.

- C. Thirdly, we assume that there exists i such that $|h(y_i)| + |h(z_i)| \neq |w_i| - m$. Let $j = \min\{i \leq k \mid |h(y_i)| + |h(z_i)| \neq |w_i| - m\}$. Then the suffixes $(a^M b^M)^M$ of g_j and f_j do not align perfectly to each other. If $|h(y_j)| + |h(z_j)| < |w_j| - m$, then the suffix $(a^M b^M)^M$ of f_j is aligned to a factor of w which starts inside w_j . This immediately causes at least M mismatches, as each group a^M will overlap to a group of which contains at least one b letter. If $|h(y_j)| + |h(z_j)| > |w_j| - m$, then the suffix $(a^M b^M)^M$ of f_j is aligned to a factor of w which starts strictly to the right of the factor w_j . However, because $M = (k\ell)^2 \gg k\ell$,

and f_j and g_j are followed by the same number of factors $(\mathbf{a}^M \mathbf{b}^M)^M$ (until the factors A_α and A_w are reached), the factor corresponding to the suffix $(\mathbf{a}^M \mathbf{b}^M)^M$ of f_j cannot start more than $k\ell$ positions to the right of w_j . It is then immediate that this factor $(\mathbf{a}^M \mathbf{b}^M)^M$ of f_j will cause at least M mismatches: each group \mathbf{a}^M will overlap to a group of which contains at least one \mathbf{b} letter.

So, from now on we can assume that the factors $(\mathbf{a}^M \mathbf{b}^M)^M$ of g_j and f_j are aligned.

- D. At this point, it is clear that in each alignment of α and w which fulfils the conditions described in items B and C: the variable x is mapped to a string of length m , and its first k occurrences are aligned to factors of the words w_1, \dots, w_k . We will now show that for each alignment of α and w in which the image of x contains a $\$$ symbol and fulfils the conditions above, there exists an alignment of α and w with at most the same number of mismatches, in which the image of x does not contain a $\$$ symbol and, once more, fulfils the conditions B and C. Assume that in our original alignment x is mapped to a string u_x of length m such that $u_x[i] = \$$. Let u_1, \dots, u_k be the factors of w_1, \dots, w_k , respectively, to which the first occurrences of the variable x are aligned. Consider the string u'_x which is obtained from u_x by simply replacing the $\$$ symbol on position i by $u_1[i]$. And then consider the alignment of α and w which is obtained from the original alignment by changing the image of x to u'_x instead of u_x . When compared to the original alignment, the new alignment has an additional mismatch caused by the occurrence of x aligned to $\m , but at least one less mismatch caused by the alignments of the first k occurrences of x . Indeed, in the original alignment, the i^{th} position of u_x was a mismatch to the i^{th} position of any string u_1, \dots, u_k , but now at least the i^{th} positions of w_1 and u'_x coincide. This shows that our claim holds. A similar argument shows that for any alignment in which x is mapped to a string containing other letters than the input letters from the CP-instance there exists an alignment in which x is mapped to a string containing only letters from the CP-instance.

Hence, from now on we can assume that the factors $(\mathbf{a}^M \mathbf{b}^M)^M$ of g_j and f_j are aligned and that the image of x has length m and is over the input alphabet of CP-instance.

Based on the observations A-D, we can show that the reduction has the desired properties. If the CP-instance admits a solution s, s_1, \dots, s_k which causes a number of mismatches less or equal to Δ , then we can produce an alignment of α to w as follows. We map x to s and, for i from 1 to k , we map x_i and y_i to the prefix of w_i occurring before s_i and, respectively, the suffix of w_i occurring after s_i . This leads to $\Delta + m$ mismatches between α and w , so the input $(w, \alpha, \Delta + m)$ of $\text{MisMatch}_{1\text{RepVar}}$ is accepted. Conversely, if we have an alignment of α and w with at most $\Delta + m$ mismatches, then we have an alignment with the same number of mismatches which fulfils the conditions summarized at the end of item D above. Hence, we can define s as the image of x in this alignment, and the strings s_1, \dots, s_k as the factors of w aligned to the first k occurrences of x from α . Clearly, for i between 1 and k , s_i is a factor of w_i . As m mismatches of the alignment were caused by the alignment of the last x to $\m , we get that $\sum_{i=1}^k d_{\text{HAM}}(s, s_i) \leq \Delta$. Thus, the instance of CP is accepted.

This concludes the proof of the correctness of our reduction. As M is clearly of polynomial size w.r.t. the size of the CP-instance, it follows that both w and α are of polynomial size $\mathcal{O}(kM^2)$. Therefore, the instance of $\text{MinMisMatch}_{1\text{RepVar}}$ can be computed in polynomial time, and our entire reduction is done in polynomial time. Moreover, we have shown that the instance $(w, \alpha, \Delta + M)$ of $\text{MinMisMatch}_{1\text{RepVar}}$ is answered positively if and only if the original instance of CP is answered positively. Finally, as the number of x blocks in α is $k + 1$, where k is the number of input strings in the instance of CP, and CP is $W[1]$ -hard with respect to this parameter, it follows that $\text{MinMisMatch}_{1\text{RepVar}}$ is also $W[1]$ -hard when the number of k -blocks in α is considered as parameter. This completes our proof. \blacktriangleleft

Note that the pattern α constructed in the reduction above is $k-1$ -local (and not k -local): a witness marking sequence is $z_1 < y_2 < z_2 < y_3 < \dots < z_{k-1} < y_k < x < y_1 < z_k$. Thus, $\text{MisMatch}_{1\text{RepVar}}$ is $W[1]$ -hard w.r.t. locality of the input pattern as well. Also, it is easy to see that $\text{scd}(\alpha) = 2$, and, by the results of [42], this shows that the treewidth of the pattern α , as defined in the same paper, is at most 3. Thus, even for classes of patterns with constant scd , number or repeated variables, or treewidth, the problems MisMatch_P and MinMisMatch_P can become intractable. In Theorem 4.6 we have shown that $\text{MinMisMatch}_{1\text{RepVar}}$ admits a polynomial time approximation scheme (for short, PTAS). We will show in the following that it does not admit an efficient PTAS (for short, EPTAS), unless $FPT = W[1]$. This means that there is no PTAS for $\text{MinMisMatch}_{1\text{RepVar}}$ such that the exponent of the polynomial in its running time is independent of the approximation ratio. To show this, we consider an optimisation variant of the problem CP, denoted minCP . In this problem, for k strings $w_1, \dots, w_k \in \Sigma^\ell$ of length ℓ and an integer $m \in \mathbb{N}$ with $m \leq \ell$, we are interested in the smallest non-negative integer Δ for which there exist strings s , of length m , and s_1, \dots, s_k , factors of length m of each w_1, \dots, w_k , respectively, such that $\sum_{i=1}^k d_{\text{HAM}}(s_i, s) = \Delta$. In [7], it is shown that minCP has no EPTAS unless $FPT = W[1]$. We can use this result and the reduction from the Theorem 4.7 to show the following result (see Appendix B).

► **Theorem 4.8.** $\text{MinMisMatch}_{1\text{RepVar}}$ has no EPTAS unless $FPT = W[1]$.

References

- 1 Amihod Amir, Moshe Lewenstein, and Ely Porat. Faster algorithms for string matching with k mismatches. *J. Algorithms*, 50(2):257–275, 2004. doi:10.1016/S0196-6774(03)00097-X.
- 2 Amihod Amir and Igor Nor. Generalized function matching. *J. Discrete Algorithms*, 5:514–523, 2007. doi:10.1016/j.jda.2006.10.001.
- 3 Dana Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21(1):46–62, 1980. doi:10.1016/0022-0000(80)90041-0.
- 4 Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *Proc. 57th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 457–466, 2016. doi:10.1109/FOCS.2016.56.
- 5 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018. doi:10.1145/2746539.2746612.
- 6 Philip Bille and Martin Farach-Colton. Fast and compact regular expression matching. *Theor. Comput. Sci.*, 409(3):486–496, 2008. doi:10.1016/j.tcs.2008.08.042.
- 7 Christina Boucher, Christine Lo, and Daniel Lokshantov. Consensus patterns (probably) has no EPTAS. In *Proc. 23rd Annual European Symposium, ESA*, volume 9294 of *Lecture Notes in Computer Science*, pages 239–250, 2015. doi:10.1007/978-3-662-48350-3_21.
- 8 Brona Brejová, Daniel G. Brown, Ian M. Harrower, Alejandro López-Ortiz, and Tomás Vinar. Sharper upper and lower bounds for an approximation scheme for consensus-pattern. In *Proc. 16th Annual Symposium Combinatorial Pattern Matching, CPM 2005*, volume 3537 of *Lecture Notes in Computer Science*, pages 1–10, 2005. doi:10.1007/11496656_1.
- 9 Brona Brejová, Daniel G. Brown, Ian M. Harrower, and Tomás Vinar. New bounds for motif finding in strong instances. In *Proc. 17th Annual Symposium Combinatorial Pattern Matching, CPM 2006*, volume 4009 of *Lecture Notes in Computer Science*, pages 94–105, 2006. doi:10.1007/11780441_10.
- 10 Karl Bringmann. Fine-grained complexity theory (tutorial). In *Proc. 36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019*, volume 126 of *LIPICs*, pages 4:1–4:7, 2019. doi:10.4230/LIPICs.STACS.2019.4.

- 11 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. 56th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 79–97, 2015. doi:10.1109/FOCS.2015.15.
- 12 Karl Bringmann and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1216–1235. SIAM, 2018. doi:10.1137/1.9781611975031.79.
- 13 Laurent Bulteau and Markus L. Schmid. Consensus strings with small maximum distance and small distance sum. *Algorithmica*, 82(5):1378–1409, 2020. doi:10.1007/s00453-019-00647-9.
- 14 Cezar Câmpeanu, Kai Salomaa, and Sheng Yu. A formal study of practical regular expressions. *Int. J. Found. Comput. Sci.*, 14:1007–1018, 2003. doi:10.1142/S012905410300214X.
- 15 Katrin Casel, Joel D. Day, Pamela Fleischmann, Tomasz Kociumaka, Florin Manea, and Markus L. Schmid. Graph and string parameters: Connections between pathwidth, cutwidth and the locality number. In *Proc. 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, volume 132 of *LIPICs*, pages 109:1–109:16, 2019. doi:10.4230/LIPICs.ICALP.2019.109.
- 16 Panagiotis Charalampopoulos, Tomasz Kociumaka, and Philip Wellnitz. Faster approximate pattern matching: A unified approach. In *Proc. 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 978–989, 2020. doi:10.1109/FOCS46700.2020.00095.
- 17 Maxime Crochemore, Christophe Hancart, and Thierry Lecroq. *Algorithms on strings*. Cambridge University Press, 2007. doi:10.1017/CB09780511546853.
- 18 Joel D. Day, Pamela Fleischmann, Florin Manea, and Dirk Nowotka. Local patterns. In *Proc. 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017*, volume 93 of *LIPICs*, pages 24:1–24:14, 2017. doi:10.4230/LIPICs.FSTTCS.2017.24.
- 19 Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Document spanners: A formal approach to information extraction. *J. ACM*, 62(2):12:1–12:51, 2015. doi:10.1145/2699442.
- 20 Michael R. Fellows, Jens Gramm, and Rolf Niedermeier. On the parameterized intractability of motif search problems. *Comb.*, 26(2):141–167, 2006. doi:10.1007/s00493-006-0011-4.
- 21 Henning Fernau, Florin Manea, Robert Mercas, and Markus L. Schmid. Revisiting Shinohara’s algorithm for computing descriptive patterns. *Theor. Comput. Sci.*, 733:44–54, 2018. doi:10.1016/j.tcs.2018.04.035.
- 22 Henning Fernau, Florin Manea, Robert Mercas, and Markus L. Schmid. Pattern matching with variables: Efficient algorithms and complexity results. *ACM Trans. Comput. Theory*, 12(1):6:1–6:37, 2020. doi:10.1145/3369935.
- 23 Henning Fernau and Markus L. Schmid. Pattern matching with variables: A multivariate complexity analysis. *Inf. Comput.*, 242:287–305, 2015. doi:10.1016/j.ic.2015.03.006.
- 24 Henning Fernau, Markus L. Schmid, and Yngve Villanger. On the parameterised complexity of string morphism problems. *Theory Comput. Syst.*, 59(1):24–51, 2016. doi:10.1007/s00224-015-9635-3.
- 25 Dominik D. Freydenberger. Extended regular expressions: Succinctness and decidability. *Theory of Comput. Syst.*, 53:159–193, 2013. doi:10.1007/s00224-012-9389-0.
- 26 Dominik D. Freydenberger. A logic for document spanners. *Theory Comput. Syst.*, 63(7):1679–1754, 2019. doi:10.1007/s00224-018-9874-1.
- 27 Dominik D. Freydenberger and Mario Holldack. Document spanners: From expressive power to decision problems. *Theory Comput. Syst.*, 62(4):854–898, 2018. doi:10.1007/s00224-017-9770-0.
- 28 Dominik D. Freydenberger and Markus L. Schmid. Deterministic regular expressions with back-references. *J. Comput. Syst. Sci.*, 105:1–39, 2019. doi:10.1016/j.jcss.2019.04.001.
- 29 Jeffrey E. F. Friedl. *Mastering Regular Expressions*. O’Reilly, Sebastopol, CA, third edition, 2006.

- 30 Pawel Gawrychowski, Florin Manea, and Stefan Siemer. Matching patterns with variables under hamming distance. *CoRR*, abs/2106.06249, 2021. [arXiv:2106.06249](https://arxiv.org/abs/2106.06249).
- 31 Pawel Gawrychowski and Przemyslaw Uznanski. Optimal trade-offs for pattern matching with k mismatches. *CoRR*, abs/1704.01311, 2017. [arXiv:1704.01311](https://arxiv.org/abs/1704.01311).
- 32 Pawel Gawrychowski and Przemyslaw Uznanski. Towards unified approximate pattern matching for hamming and l_1 distance. In *Proc. 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPICs*, pages 62:1–62:13, 2018. doi:10.4230/LIPICs.ICALP.2018.62.
- 33 Juha Kärkkäinen and Peter Sanders. Simple linear work suffix array construction. In *Proc. 30th International Colloquium Automata, Languages and Programming, ICALP 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 943–955, 2003. doi:10.1007/3-540-45061-0_73.
- 34 Juha Kärkkäinen, Peter Sanders, and Stefan Burkhardt. Linear work suffix array construction. *J. ACM*, 53(6):918–936, 2006. doi:10.1145/1217856.1217858.
- 35 Gad M. Landau and Uzi Vishkin. Efficient string matching in the presence of errors. In *Proc. 26th Annual Symposium on Foundations of Computer Science, FOCS 1985*, pages 126–136, 1985. doi:10.1109/SFCS.1985.22.
- 36 Ming Li, Bin Ma, and Lusheng Wang. Finding similar regions in many sequences. *J. Comput. Syst. Sci.*, 65(1):73–96, 2002. doi:10.1006/jcss.2002.1823.
- 37 M. Lothaire. *Combinatorics on Words*. Cambridge University Press, 1997. doi:10.1017/CB09780511566097.
- 38 M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002. doi:10.1017/CB09781107326019.
- 39 Florin Manea and Markus L. Schmid. Matching patterns with variables. In *Proc. 12th International Conference Combinatorics on Words, WORDS 2019*, volume 11682 of *Lecture Notes in Computer Science*, pages 1–27, 2019. doi:10.1007/978-3-030-28796-2_1.
- 40 Dániel Marx. Closest substring problems with small distances. *SIAM J. Comput.*, 38(4):1382–1410, 2008. doi:10.1137/060673898.
- 41 Eugene W. Myers and Webb Miller. Approximate matching of regular expressions. *Bull. Math. Biol.*, 51(1):5–37, 1989. doi:10.1007/BF02458834.
- 42 Daniel Reidenbach and Markus L. Schmid. Patterns with bounded treewidth. *Inf. Comput.*, 239:87–99, 2014. doi:10.1016/j.ic.2014.08.010.
- 43 Markus L. Schmid. A note on the complexity of matching patterns with variables. *Inf. Process. Lett.*, 113(19):729–733, 2013. doi:10.1016/j.ipl.2013.06.011.
- 44 Markus L. Schmid and Nicole Schweikardt. A purely regular approach to non-regular core spanners. In *Proc. 24th International Conference on Database Theory, ICDT 2021*, volume 186 of *LIPICs*, pages 4:1–4:19, 2021. doi:10.4230/LIPICs.ICDT.2021.4.
- 45 Takeshi Shinohara. Polynomial time inference of pattern languages and its application. In *Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 191–209, 1982.
- 46 Takeshi Shinohara and Setsuo Arikawa. Pattern inference. In *Algorithmic Learning for Knowledge-Based Systems, GOSLER Final Report*, volume 961 of *LNAI*, pages 259–291, 1995.
- 47 Przemyslaw Uznanski. Recent advances in text-to-pattern distance algorithms. In *Proc. 16th Conference on Computability in Europe, CiE 2020*, volume 12098 of *Lecture Notes in Computer Science*, pages 353–365, 2020. doi:10.1007/978-3-030-51466-2_32.
- 48 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

A Computational Model

The computational model we use to describe our results is the standard unit-cost RAM with logarithmic word size: for an input of size n , each memory word can hold $\log n$ bits. Arithmetic and bitwise operations with numbers in $[1 : n]$ are, thus, assumed to take $O(1)$

time. Numbers larger than n , with ℓ bits, are represented in $O(\ell/\log n)$ memory words, and working with them takes time proportional to the number of memory words on which they are represented. In all the problems, we assume that we are given a word w and a pattern α , with $|w| = n$ and $|\alpha| = m \leq n$, over a terminal-alphabet $\Sigma = \{1, 2, \dots, \sigma\}$, with $|\Sigma| = \sigma \leq n$. The variables are chosen from the set $\{x_1, \dots, x_n\}$ and can be encoded as integers between $n + 1$ and $2n$. That is, we assume that the processed words are sequences of integers (called letters or symbols), each fitting in $O(1)$ memory words. This is a common assumption in string algorithms: the input alphabet is said to be *an integer alphabet*. For instance, the same assumption was also used for developing efficient algorithms for `Match` in [21]. For a more detailed general discussion on this model see, e.g., [17].

B Proofs

► **Lemma 3.1.** *Let w and u , with $|w| = |u| = n$, be two words and δ a non-negative integer. Assume that, in a preprocessing phase, we have constructed $LCS_{w,u}$ -data structures. We can compute $\min(\delta + 1, d_{\text{HAM}}(u, w))$ using $\delta + 1$ $LCS_{w,u}$ queries, so in $O(\delta)$ time.*

Proof. Let $a = b = m$ and $d = 0$. While $a > 0$ and $d \leq \delta$ execute the following steps. Compute $h = LCS_{w,u}(a, b)$. If $h < b$, then increment d by 1, set $a \leftarrow a - h - 1$ and $b \leftarrow b - h - 1$, and start another iteration of the while-loop. If $h = b$, then set $b \leftarrow 0$ and exit the while-loop.

It is not hard to note that before each iteration of the while loop it holds that $d = d_{\text{HAM}}(w[a + 1 : m], u[b + 1 : m])$. When the while loop is finished, $d = \min(d_{\text{HAM}}(w[i - m + 1 : i], u[1 : m]), \delta + 1)$. In each iteration we first identify the length h of the longest common suffix of $w[1 : a]$ and $u[1 : b]$. Then, we jump over this suffix, as it causes no mismatches, and have either traversed completely the words w and u (and we do not need to do anything more), or we have reached a mismatch between w and u , on position $a - h = b - h$. In the latter case, we count this mismatch, jump over it, and repeat the process (but only if the number of mismatches is still at most δ). So, in other words, we go through the mismatches of w and u , from right to left, and jump from one to the next one using $LCS_{w,u}$ queries. If we have more than δ mismatches, we do not count all of them, but stop as soon as we have met the $(\delta + 1)^{\text{th}}$ mismatch. Accordingly, the algorithm is correct. Clearly, we only need $\delta + 1$ $LCS_{w,u}$ -queries and the time complexity of this algorithm is $O(\delta)$, once the $LCS_{w,u}$ -data structures are constructed. ◀

► **Lemma 3.2.** *Given a word w , with $|w| = n$, a word u , with $|u| = m < n$, and a non-negative integer δ , we can compute in $O(n\delta)$ time the array $D[m : n]$ with $n - m + 1$ elements, where $D[i] = \min(\delta + 1, d_{\text{HAM}}(w[i - m + 1 : i], u))$.*

Proof. We first construct, in linear time, the $LCS_{w,u}$ -data structures for the input words. Note that the $LCS_{w,u}$ -data structure can be directly used as $LCS_{w[i:i+m-1],u}$ data structure, for all $i \leq n - m + 1$.

Then, for each position i of w , with $i \leq m$, we use Lemma 3.1 to compute, in $O(\delta)$ time the value $d = \min(d_{\text{HAM}}(u, w[i - m + 1 : i]), \delta + 1)$. We then set $D[i] \leftarrow d$. By the correctness of Lemma 3.1, we get the correctness of this algorithm. Clearly, its time complexity is $O(n\delta)$. ◀

► **Theorem 4.4.** *$\text{MinMisMatch}_{1\text{RepVar}}$ and $\text{MisMatch}_{1\text{RepVar}}$ can be solved in $O(n^{k+2}m)$ time, where k is the number of x -blocks in the input pattern α .*

Proof. Once more, we only show how $\text{MinMismatch}_{1\text{RepVar}}$ can be solved. The result for $\text{Mismatch}_{1\text{RepVar}}$ follows then immediately.

In $\text{MinMismatch}_{1\text{RepVar}}$, we are given a word w , of length n , and a pattern α , of length m , which, as stated above, has exactly k x -blocks. Thus $\alpha = \prod_{i=1}^k (\gamma_{i-1}\beta_i)\gamma_k$, where the factors β_i , for $i \in \{1, \dots, k\}$, are the x -blocks of α . It is easy to observe that $\text{var}(\gamma_i) \cap \text{var}(\gamma_j) = \emptyset$, for all i and j , and $\gamma = \gamma_0\gamma_1 \cdots \gamma_k$ is a regular pattern.

When aligning α to w we actually align each of the patterns γ_j and β_i , for $0 \leq j \leq k$ and $1 \leq i \leq k$, to respective factors of the word w . Moreover, the factors to which these patterns are respectively aligned are completely determined by the length ℓ of the image of x , and the starting positions h_i of the factors aligned to the patterns β_i , for $1 \leq i \leq k$. Knowing the length ℓ of the image of x , we can also compute, for $1 \leq i \leq k$, the length ℓ_i of β_i , when x is replaced by a string of length ℓ . In this case, γ_0 is aligned $u_0 = w[1..h_1 - 1]$ and, for $1 \leq i \leq k$, β_i is aligned to $w_i = w[h_i : h_i + \ell_i - 1]$ and γ_i is aligned $u_i = w[h_i + \ell_i : h_{i+1} - 1]$ (where $h_{k+1} = n + 1$). Thus, $\beta_1 \cdots \beta_k$ matches $w_1 \cdots w_k$ and we can use Theorem 4.2 to determine $d_{\text{HAM}}(\beta_1 \cdots \beta_k, w_1 \cdots w_k)$ (or, in other words, determine the string u_x that should replace x in order to realize this Hamming distance). Further, we can use Theorem 3.4 to compute $d_{\text{HAM}}(\gamma_i, u_i)$, for all $i \in \{0, \dots, k\}$. Adding all these distances up, we obtain a total distance $D_{\ell, h_1, \dots, h_k}$; this value depends on ℓ, h_1, \dots, h_k .

So, we can simply iterate over all possible choices for ℓ, h_1, \dots, h_k and find $d_{\text{HAM}}(\alpha, w)$ as the minimum of the numbers $D_{\ell, h_1, \dots, h_k}$.

By the explanations above, it is straightforward that the approach is correct: we simply try all possibilities of aligning α with w . The time complexity is, for each choice of ℓ, h_1, \dots, h_k , $O(\sum_{i=1}^k |w_i|) \subseteq O(n)$ for the part corresponding to the computation of the optimal alignment between the factors β_i and the words w_i , and $O(\sum_{i=0}^k |u_i| d_{\text{HAM}}(\gamma_i, u_i)) \subseteq O(nm)$ for the part corresponding to the computation of the optimal alignment between the factors γ_i and the words u_i . So, the overall complexity of this algorithm is $O(n^{k+2}m)$. ◀

► **Theorem 4.5.** $\text{MinMismatch}_{\text{kLOC}}$ and $\text{Mismatch}_{\text{kLOC}}$ can be solved in $O(n^{2k+2}m)$ time.

Proof. We only present the solution for $\text{MinMismatch}_{\text{kLOC}}$ (as it trivially works in the case of $\text{Mismatch}_{\text{kLOC}}$ too).

Let us note that, by the results in [18], we can compute a marking sequence of α in $O(m^{2k}k)$ time. So, after such a preprocessing phase, we can assume that we have a word w , a k -local pattern α (with p variables) with a witness marking sequence $x_1 \leq \dots \leq x_p$ for the k -locality of α , and we want to compute $d_{\text{HAM}}(\alpha, w)$.

Generally, the main idea behind matching kLOC -patterns is that when looking for possible ways to align such a pattern α to a word w we can consider the variables in the order given by the marking sequence, and, when reaching variable x_i , we try all possible assignments for x_i . The critical observation here is that after each such assignment of a new variable, we only need to keep track of the way the $t \leq k$ length-maximal factors of α , which contain only marked variables and terminals, match (at most) $t \leq k$ factors of w .

We will use this approach in our algorithm for $\text{MinMismatch}_{\text{kLOC}}$.

The first step of this algorithm is the following. We go through α and identify all x_1 -blocks: $\beta_{1,1}, \dots, \beta_{1,j_1}$. Because α is k -local, we have that $j_1 \leq k$. For each $2j_1$ -tuple (i_1, \dots, i_{2j_1}) of positions of w , we compute the minimum number of mismatches if we align (simultaneously) the patterns β_g to the factors $w[i_{2g-1} : i_{2g}]$, for g from 1 to j_1 , respectively. This reduces to finding an assignment for x_1 which aligns optimally the patterns $\beta_{1,g}$ to the respective factors, and can be done in $O(n)$ time using Theorem 4.2. For each $2j_1$ -tuple (i_1, \dots, i_{2j_1}) of

positions of w , we denote by $M_1(i_1, \dots, i_{2j_1})$ the minimum number of mismatches resulting from the (simultaneous) alignment of the patterns $\beta_{1,g}$ to the factors $w[i_{2g-1} : i_{2g}]$, for g from 1 to j_1 , respectively. Clearly, M_1 can be seen as a j_1 -dimensional array.

Assume that after $h \geq 1$ steps of our algorithm we have computed the factors $\beta_{h,1}, \dots, \beta_{h,j_h}$ of α , which are length-maximal factors of α which only contain the variables x_1, \dots, x_h and terminals (i.e., extending them to the left or right would introduce a new variable x_ℓ with $\ell > h$); as α is k -local, we have $j_h \leq k$. Moreover, for each $2j_h$ -tuple (i_1, \dots, i_{2j_h}) of positions of w , we have computed $M_h(i_1, \dots, i_{2j_h})$, the minimum number of mismatches if we align (simultaneously) the patterns $\beta_{h,g}$ to the factors $w[i_{2g-1} : i_{2g}]$, for g from 1 to j_h , respectively. M_h is implemented as a j_h dimensional array, and this assumption clearly holds after the first step.

We now explain how step $h + 1$ is performed.

1. We compute the factors $\beta_{h+1,1}, \dots, \beta_{h+1,j_{h+1}}$ of α , which are length-maximal factors of α which only contain the variables x_1, \dots, x_{h+1} and terminals (i.e., extending them to the left or right would introduce a new variable x_ℓ with $\ell > h + 1$). Clearly, $\beta_{h+1,r}$ is either an x_{h+1} -block or it has the form $\beta_{h+1,r} = \gamma_{r,0}\beta_{h,a_r}\gamma_{r,1} \cdots \beta_{r,a_r+b_r}\gamma_{r,b_r+1}$ where the patterns $\gamma_{r,t}$ contain only the variable x_{h+1} and terminals and extending $\beta_{h+1,r}$ to the left or right would introduce a new variable x_ℓ with $\ell > h + 1$.
2. We initialize the values $M_{h+1}(i_1, \dots, i_{2j_{h+1}}) \leftarrow \infty$, for each $2j_{h+1}$ -tuple $(i_1, \dots, i_{2j_{h+1}})$ of positions of w .
3. For each $\ell \leq n$ (where ℓ corresponds to the length of the image of x_{h+1}) and each $2j_h$ -tuple (i_1, \dots, i_{2j_h}) of positions of w such that $M_h(i_1, \dots, i_{2j_h})$ is finite do the following:
 - a. We compute the tuple $(i'_1, \dots, i'_{2j_{h+1}})$ such that $\beta_{h+1,g}$ is aligned to the factor $w[i'_{2g-1} : i'_{2g}]$, for g from 1 to j_{h+1} , respectively. This can be computed based on the fact that the factors $\beta_{h,g}$ are aligned to the factors $w[i_{2g-1} : i_{2g}]$, for g from 1 to j_h , respectively, and the image of x_{h+1} has length ℓ .
 - b. We compute the factors of w aligned to x_{h+1} in the alignment computed in the previous line. Then, we can use the algorithm from Theorem 4.2 and the value of $M_h(i_1, \dots, i_{2j_h})$ to compute an assignment for x_{h+1} which aligns optimally the patterns $\beta_{h+1,g}$ to the corresponding factors of w .
 - c. If the number of the mismatches in this alignment is smaller than the current value of $M_{h+1}(i'_1, \dots, i'_{2j_{h+1}})$, we update $M_{h+1}(i'_1, \dots, i'_{2j_{h+1}})$.

This dynamic programming approach is clearly correct. In $M_{h+1}(i_1, \dots, i_{2j_{h+1}})$ we have the optimal alignment of the patterns $\beta_{h+1,1}, \dots, \beta_{h+1,j_{h+1}}$ to $w[i_1 : i_2], \dots, w[i_{2j_{h+1}-1} : i_{2j_{h+1}}]$, respectively. As far as the complexity is concerned, the lines 1, 3.a, 3.b, 3.c can be implemented in linear time, while the for-loop is iterated $O(n^{2k+1})$ times. Line 2 takes $O(n^{2k})$ times. The whole computation in step $h + 1$ of the algorithm takes, thus, $O(n^{2k+1})$ time.

Now, we execute the procedure described above for h from 2 to m , and, in the end, we compute the array M_m . The answer to our instance of the problem $\text{MinMismatch}_{\text{KLOC}}$ is $M_m(1, n)$. The overall time complexity needed to perform this computation is $O(mn^{2k+1})$ time. The preprocessing phase, in which the marking sequence and the array M_1 were computed, takes also $O(mn^{2k+1})$ time. So, the complexity stated in the statement is reached by our algorithm. \blacktriangleleft

► Theorem 4.6. *For each constant $r \geq 3$, there exists an algorithm with run-time $O(n^{r+3})$ for $\text{MinMismatch}_{1\text{RepVar}}$ whose output distance is at most $\min \left\{ 2, \left(1 + \frac{4\sigma-4}{\sqrt{e}(\sqrt{4r+1}-3)} \right) \right\} d_{\text{HAM}}(\alpha, w)$.*

Proof. We first note that there exists a relatively simple algorithm solving $\text{MinMisMatch}_{1\text{RepVar}}$ such that the output distance is no more than $2d_{\text{HAM}}(\alpha, w)$ (which also works for integer alphabets).

Indeed, assume that we have a substitution h for which $d_{\text{HAM}}(h(\alpha), w) = d_{\text{HAM}}(\alpha, w)$. Assume that the repeated variable x is mapped by h to a string u and the t occurrences of x are aligned, under h , to the factors w_1, w_2, \dots, w_t of w . Now, let w_i be such $d_{\text{HAM}}(u, w_i) \leq d_{\text{HAM}}(u, w_j)$ for all $j \neq i$. Let us consider now the substitution h' which substitutes x by w_i and all the other variables exactly as h did. We claim that $d_{\text{HAM}}(h'(\alpha), u) \leq 2d_{\text{HAM}}(h(\alpha), u)$. It is easy to see that $d_{\text{HAM}}(h'(\alpha), w) - d_{\text{HAM}}(h(\alpha), w) = \sum_{j=i}^t (d_{\text{HAM}}(w_i, w_j) - d_{\text{HAM}}(u, w_j)) \leq \sum_{j=i}^t (d_{\text{HAM}}(w_i, u) + d_{\text{HAM}}(u, w_j) - d_{\text{HAM}}(u, w_i))$ (where the last inequality follows from the triangle inequality for the Hamming distance). Thus, $d_{\text{HAM}}(h'(\alpha), w) - d_{\text{HAM}}(h(\alpha), w) \leq \sum_{j=i}^t d_{\text{HAM}}(w_i, u) \leq \sum_{j=i}^t d_{\text{HAM}}(w_j, u) \leq d_{\text{HAM}}(h(\alpha), u)$. So our claim holds.

A consequence of the previous observation is that there exists a substitution h' that maps x to a factor of w and produces a string $h'(\alpha)$ such that $d_{\text{HAM}}(h'(\alpha), u) \leq 2d_{\text{HAM}}(\alpha, u)$. So, for each factor u of w , we x by u in α to obtain a regular pattern α' , then use Theorem 3.4 to compute $d_{\text{HAM}}(\alpha', w)$. We return the smallest value $d_{\text{HAM}}(\alpha', w)$ achieved in this way. Clearly, this is at most $2d_{\text{HAM}}(\alpha, w)$. The complexity of this algorithm is $O(n^4)$, as it simply uses the quadratic algorithm of Theorem 3.4 for each factor of w .

We will now show how this algorithm can be modified to produce a value closer to $d_{\text{HAM}}(\alpha, w)$, while being less efficient.

The algorithm consists of the following main steps:

1. For $\ell \leq n/r$ and r factors u_1, \dots, u_r of length ℓ of w do the following:
 - a. Compute u_{u_1, \dots, u_r} the median string of u_1, \dots, u_r using Lemma 4.1.
 - b. Let α' be the regular pattern obtained by replacing x by u_{u_1, \dots, u_r} in α .
 - c. Compute the distance $d_{u_1, \dots, u_r} = d_{\text{HAM}}(\alpha', w)$ using Theorem 3.4.
2. Return the smallest distance d_{u_1, \dots, u_r} computed in the loop above.

Clearly, for $r = 1$ the above algorithm corresponds to the simple algorithm presented in the beginning of this proof. Let us analyse its performance for an arbitrary choice of r .

The complexity is easy to compute: we need to consider all possible choices for ℓ and the starting positions of u_1, \dots, u_r . So, we have $O(n^{r+1})$ possibilities to select the non-overlapping factors u_1, \dots, u_r of length ℓ of w . The computation done inside the loop can be performed in $O(n^2)$ time. So, overall, our algorithm runs in $O(n^{r+3})$ time.

Now, we want to estimate how far away from $d_{\text{HAM}}(\alpha, w)$ is the value this algorithm returns. In this case, we will make use of the fact that the input terminal-alphabet is constant. We follow closely (and adapt to our setting) the approach from [36].

Firstly, a notation. In step 1.b of the algorithm above, we align α' to w with a minimal number of mismatches. In this alignment, let d'_{u_1, \dots, u_r} be the total number of mismatches caused by the factors u_{u_1, \dots, u_r} which replaced the occurrences of the variable x in α .

Now, assume that we have a substitution h for which $d_{\text{HAM}}(h(\alpha), w) = d_{\text{HAM}}(\alpha, w) = d_{\text{opt}}$. Assume also that the repeated variable x is mapped by h to a string u_{opt} of length L and the t occurrences of x are aligned, under h , to the factors w_1, w_2, \dots, w_t of w . Let d'_{opt} be the number of mismatches caused by the alignment of the images of the t occurrences of x under h to the factors w_1, w_2, \dots, w_t . Finally, let $\rho = 1 + \frac{4\sigma-4}{\sqrt{e}(\sqrt{4r+1}-3)}$.

Note that, for $\ell = L$, u_1, \dots, u_r correspond to a set of randomly chosen numbers i_1, \dots, i_r from $\{1, \dots, n\}$: their starting positions. We will show in the following that $E[d'_{u_1, \dots, u_r}] \leq \rho d'_{\text{opt}}$. If this inequality holds, then we can apply the probabilistic method: there exists at least a choice of u_1, \dots, u_r of length L such that $d'_{u_1, \dots, u_r} \leq \rho d'_{\text{opt}}$. As we try all possible lengths ℓ and all variants for choosing u_1, \dots, u_r of length ℓ , we will also consider

the choice of u_1, \dots, u_r of length L such that $d'_{u_1, \dots, u_r} \leq \rho d'_{opt}$, and it is immediate that, for that, for the respective u_1, \dots, u_r we also have that $d_{u_1, \dots, u_r} \leq \rho d_{opt}$. Thus, the value returned by our algorithm is at most ρd_{opt} .

So, let us show the inequality $E[d'_{u_1, \dots, u_r}] \leq \rho d'_{opt}$.

For $\mathbf{a} \in \Sigma$, let $f_j(\mathbf{a}) = |\{i \mid 1 \leq i \leq t, w_i[j] = \mathbf{a}\}|$. Now, for an arbitrary string s of length L , we have that $\sum_{i=1}^t d_{\text{HAM}}(w_i, s) = \sum_{j=1}^L (t - f_j(s[j]))$. So, for $s = u_{opt}$ we get $\sum_{i=1}^t d_{\text{HAM}}(w_i, u_{opt}) = \sum_{j=1}^L (t - f_j(u_{opt}[j]))$, and for $s = u_{u_1, \dots, u_r}$ we have that $d'_{opt} = \sum_{i=1}^t d_{\text{HAM}}(w_i, u_{u_1, \dots, u_r}) = \sum_{j=1}^L (t - f_j(u_{u_1, \dots, u_r}[j]))$.

Therefore, $E[d'_{u_1, \dots, u_r}] = E\left[\sum_{j=1}^L (t - f_j(u_{u_1, \dots, u_r}[j]))\right] = \sum_{j=1}^L E[t - f_j(u_{u_1, \dots, u_r}[j])]$.

Consequently, $E[d'_{u_1, \dots, u_r} - d'_{opt}] = \sum_{j=1}^L (E[t - f_j(u_{u_1, \dots, u_r}[j])] - t + f_j(u_{opt}[j]))$.

That is, $E[d'_{u_1, \dots, u_r} - d'_{opt}] = \sum_{j=1}^L E[f_j(u_{opt}[j]) - f_j(u_{u_1, \dots, u_r}[j])]$.

By Lemma 7 of [36], we have that $E[f_j(u_{opt}[j]) - f_j(u_{u_1, \dots, u_r}[j])] \leq (\rho - 1)(t - f_j(u_{opt}[j]))$.

Hence, $E[d'_{u_1, \dots, u_r} - d'_{opt}] \leq (\rho - 1) \sum_{j=1}^L (t - f_j(u_{opt}[j])) = (\rho - 1)d'_{opt}$.

So, we indeed have that $E[d'_{u_1, \dots, u_r}] \leq \rho d'_{opt}$.

In conclusion, the statement of the theorem holds. \blacktriangleleft

► **Theorem 4.8.** $\text{MinMisMatch}_{1\text{RepVar}}$ has no EPTAS unless $FPT = W[1]$.

Proof. Assume, for the sake of a contradiction, that $\text{MinMisMatch}_{1\text{RepVar}}$ has an EPTAS. That is, for an input word w and an 1RepVar -pattern α , there exists a polynomial time algorithm which returns as answer to $\text{MinMisMatch}_{1\text{RepVar}}$ a value $\delta' \leq (1 + \epsilon)d_{\text{HAM}}(\alpha, w)$, and the exponent of the polynomial in its running time is independent of ϵ .

An algorithm for minCP would first implement the reduction in Theorem 4.7 to obtain a word w and a pattern α . Then it uses the EPTAS for $\text{MinMisMatch}_{1\text{RepVar}}$ to approximate the distance between α and w with approximation ratio $(1 + \frac{\epsilon}{2m})$. Assuming that this EPTAS returns the value D , the answer returned by this algorithm for the minCP problem is $D - m$.

As explained in the proof of Theorem 4.7, it is easy to see that the distance between the word w and the pattern α constructed in the respective reduction is $m + \Delta$, if Δ is the answer to the instance of the minCP problem. Thus, the value D returned by the EPTAS for $\text{MinMisMatch}_{1\text{RepVar}}$ fulfils $m + \Delta \leq D \leq (1 + \frac{\epsilon}{2m})(m + \Delta)$. So, we have $\Delta \leq D - m \leq \frac{\epsilon}{2} + (1 + \frac{\epsilon}{2m})\Delta$. We get that $\Delta \leq D - m \leq (1 + \frac{\epsilon}{2m} + \frac{\epsilon}{2\Delta})\Delta \leq (1 + \epsilon)\Delta$. So, indeed, $D - m$ would be a $(1 + \epsilon)$ -approximation of Δ .

Therefore, this would yield an EPTAS for minCP . This is a contradiction to the results reported in [7], where it was shown that such an EPTAS does not exist, unless $FPT = W[1]$. This concludes our proof. \blacktriangleleft

Keyboards as a New Model of Computation

Yoan Gérard ✉

ENS Paris-Saclay, France

Bastien Laboureix ✉

ENS Paris-Saclay, France

Corto Mascle ✉

ENS Paris-Saclay, France

Valentin D. Richard ✉ 

ENS Paris-Saclay, France

Abstract

We introduce a new formalisation of language computation, called keyboards. We consider a set of atomic operations (writing a letter, erasing a letter, going to the right or to the left) and we define a keyboard as a set of finite sequences of such operations, called keys. The generated language is the set of words obtained by applying some non-empty sequence of those keys. Unlike classical models of computation, every key can be applied anytime. We define various classes of languages based on different sets of atomic operations, and compare their expressive powers. We also compare them to rational, context-free and context-sensitive languages. We obtain a strict hierarchy of classes, whose expressiveness is orthogonal to the one of the aforementioned classical models. We also study closure properties of those classes, as well as fundamental complexity problems on keyboards.

2012 ACM Subject Classification Theory of computation

Keywords and phrases formal languages, models of computation, automata theory

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.49

Related Version *Full version in both French and English:* <https://arxiv.org/abs/2102.10182>

Acknowledgements We want to thank Pierre Béaur, Lucas Buéri, Jean-Baptiste Daval, Paul Gastin, Colin Geniet, Valentin Maestracci and Clément Théron for their helpful advice and feedback.

1 Introduction

We present a new formalisation of languages, called keyboards. A keyboard K is a finite set of keys, which are finite sequences of atomic operations, such as writing or erasing a letter, going one position to the right or to the left... The language of K is the set of words obtained by applying a sequence of its keys on an initially empty writing space. The idea comes from the image of a malfunctioning keyboard, in which a key does a sequence of operations instead of just one (for instance the key 'a' writes b then c , and then goes to the left).

Studying the set generated by a set of algebraic operations is far from new: many works exist on the sets generated by a subset of elements of an algebraic structure, for instance in the context of semigroup and group theory [2, 7], of matrix monoids [1, 8] or the theory of codes [3]. There is however, to the best of the author's knowledge, no previous work on a model resembling the one presented here.

The atomic operations we use in this paper are the base of other models of computation, such as forgetting automata and erasing automata [4, 5, 9]. The use of those operations was originally to simulate some analysis strategies in linguistics. As a first study of the model, we chose the actions of the operations (backspace and arrows) to behave like an actual keyboard in a text editor.



© Yoan Gérard, Bastien Laboureix, Corto Mascle, and Valentin D. Richard;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 49; pp. 49:1–49:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We can define various classes of languages based on the set of atomic operations we consider, and compare their expressive powers between them, and to well-known classes of languages. We obtain a strict hierarchy of classes, with a wide range of expressiveness and difficulty of comprehension. The expressiveness of keyboards seems to be overall orthogonal to the ones of classical models of computation, which we explain by two key differences with the latter.

First, keyboards are blind and memoryless, in that they do not have states and cannot read the tape at any point. Second, because of this weakness, we can allow operations such as moving in the word or erasing letters without blowing up their expressive power too much.

The main interests of keyboards are: 1. to obtain many deep and complex mathematical questions from a deceptively simple model, and 2. that their expressiveness is very different from the ones of classical models. A language that is simple to express with a keyboard may be more complicated for automata, and vice versa. This paper is meant as a first step in the study and comprehension of keyboards and their languages.

The paper is organised as follows. In Sections 2 and 3 we establish notations and basic definitions. Section 4 and 5 are dedicated to building properties and tools necessary to the study of keyboards. In Section 6 we dive into the specific properties of each keyboard class, and prove the inclusions of some of them in regular, context-free and context-sensitive languages. Then in Section 7, we study the inclusions between those classes, in particular showing that they are all different. Some complexity results are given in Section 8. Finally, in Section 9 we show that keyboard classes are not stable by union or intersection, and that some (but not all) of them are stable by mirror.

We do not provide full proofs, but only sketch of proofs for some results. The technical details can be found in the full version.

2 Preliminaries

Given a finite alphabet A , we note A^* the set of finite words over A and A^+ for the set of non-empty ones. Given a word $w = a_1 \cdots a_n \in A^*$, we write $|w|$ for its length and, for all $a \in A$, $|w|_a$ the number of occurrences of a in w . For all $1 \leq i, j \leq n$ we use the notation $w[i]$ for the i^{th} letter of w (i.e. a_i) and $w[i, j]$ for its factor $a_i \cdots a_j$ (and ε if $j < i$). We denote the mirror of w by $\tilde{w} = a_n \cdots a_1$.

We write $\text{Pref}(w)$ for the set of prefixes of w , $\text{Suff}(w)$ for its set of suffixes, $\text{Fact}(w)$ for its set of factors and $\text{Sub}(w)$ for its set of subwords.

We represent a finite automaton A as a tuple $(Q, \Delta, \text{Init}, \text{Fin})$ with Q a finite set of states, $\Delta : Q \times A \rightarrow 2^Q$ a transition function, and $\text{Init}, \text{Fin} \subseteq Q$ sets of initial and final states.

We represent a pushdown automaton on A as a tuple $(Q, \Gamma, \perp, \Delta, \text{Init}, \text{Fin})$ with

- Q a finite set of states ;
- Γ a finite stack alphabet ;
- $\perp \in \Gamma$ an initial stack symbol ;
- $\Delta : Q \times A \times (\Gamma \cup \{-\})^2 \rightarrow 2^Q$ a transition function ;
- Init and Fin sets of initial and final states.

We accept a word on final states with an empty stack. We write transitions as follows:

$$s_1 \xrightarrow[\text{op}_1, \text{op}_2]{a} s_2$$

with

- $\text{op}_1 = \uparrow\gamma$ if we pop $\gamma \in \Gamma$, and $\text{op}_1 = -$ if we do not pop anything.
- $\text{op}_2 = \downarrow\gamma$ if we push $\gamma \in \Gamma$ on the stack, and $\text{op}_2 = -$ if we do not push anything.

We will use ε -transitions in both finite and pushdown automata to simplify some proofs.

For more details and properties of those models, we refer the reader to [6].

3 Definitions

We fix a finite alphabet A and the following special symbols, taken out of A :

The backspace : \leftarrow The left arrow : \blacktriangleleft The right arrow : \blacktriangleright

The set of all symbols is $S \triangleq A \cup \{\leftarrow, \blacktriangleleft, \blacktriangleright\}$. An element of S is called an *atomic operation*.

► **Definition 1.** A configuration is a pair of words $(u, v) \in A^* \times A^*$. We will use $\mathcal{C}(A)$ to denote the set of configurations over A , and $\langle u|v \rangle$ to denote the configuration (u, v) .

We define the notation $\langle u|v \rangle_i$ as the letter at position i in the configuration with respect to the cursor: $\langle u|v \rangle_i = \tilde{u}[-i]$ if $i < 0$ and $v[i]$ if $i > 0$.

► **Definition 2.** The action of an atomic operation $\sigma \in S$ on a configuration $\langle u|v \rangle$ is written $\langle u|v \rangle \cdot \sigma$ and is defined as follows:

$$\begin{aligned} \langle u|v \rangle \cdot a &= \langle ua|v \rangle \text{ if } a \in A. \\ \langle \varepsilon|v \rangle \cdot \leftarrow &= \langle \varepsilon|v \rangle \quad \text{and} \quad \langle u'a|v \rangle \cdot \leftarrow = \langle u'|v \rangle \\ \langle \varepsilon|v \rangle \cdot \blacktriangleleft &= \langle \varepsilon|v \rangle \quad \text{and} \quad \langle u'a|v \rangle \cdot \blacktriangleleft = \langle u'|av \rangle \\ \langle u|\varepsilon \rangle \cdot \blacktriangleright &= \langle u|\varepsilon \rangle \quad \text{and} \quad \langle u|av' \rangle \cdot \blacktriangleright = \langle ua|v' \rangle \end{aligned}$$

We will sometimes write $\langle u|v \rangle \xrightarrow{\sigma} \langle u'|v' \rangle$ for $\langle u'|v' \rangle = \langle u|v \rangle \cdot \sigma$.

► **Example 3.** By applying the following sequence of atomic operations $\leftarrow, a, \blacktriangleright, \blacktriangleright, b$ to the configuration $\langle c|d \rangle$, we obtain the following rewriting derivation:

$$\langle c|d \rangle \xrightarrow{\leftarrow} \langle \varepsilon|d \rangle \xrightarrow{a} \langle a|d \rangle \xrightarrow{\blacktriangleright} \langle ad|\varepsilon \rangle \xrightarrow{\blacktriangleright} \langle ad|\varepsilon \rangle \xrightarrow{b} \langle adb|\varepsilon \rangle.$$

► **Definition 4.** We define other semantics for atomic operations, called effective semantics. The difference with the previous ones is that we forbid application of atomic operations without effect (such as backspace when the left word of the configuration is empty). Formally, given $u, v \in A^*, a \in A$ we have:

$$\begin{aligned} \langle u|v \rangle &\xrightarrow{a}_e \langle ua|v \rangle & \langle u'a|v \rangle &\xrightarrow{\leftarrow}_e \langle u'|v \rangle \\ \langle u'a|v \rangle &\xrightarrow{\blacktriangleleft}_e \langle u'|av \rangle & \langle u|av' \rangle &\xrightarrow{\blacktriangleright}_e \langle ua|v' \rangle \end{aligned}$$

We also define the operator \odot by $\langle u|v \rangle \odot \sigma = \langle u'|v' \rangle$ if and only if $\langle u|v \rangle \xrightarrow{\sigma}_e \langle u'|v' \rangle$.

► **Definition 5.** A key is a sequence of atomic operations, seen as a word on S . We will use $\mathcal{T}(S)$ to denote the set of keys on S (variables k, t, \dots), or \mathcal{T} if there is no ambiguity.

► **Definition 6.** The action of a key over a configuration is defined inductively as follows:

$$\begin{cases} \langle u|v \rangle \cdot \varepsilon = \langle u|v \rangle \\ \langle u|v \rangle \cdot (\sigma t) = (\langle u|v \rangle \cdot \sigma) \cdot t \end{cases}$$

We extend the notation $\langle u|v \rangle \xrightarrow{t} \langle u'|v' \rangle$ to keys. We define $\langle u|v \rangle \xrightarrow{t}_e \langle u'|v' \rangle$ and $\langle u|v \rangle \odot t$ analogously.

► **Remark 7.** We will also consider sequences of keys $\tau = t_1 \dots t_n$. The action of τ is obtained by composing the actions of the t_i , hence applying τ has the same effect as applying sequentially t_1, \dots, t_n . Note that τ is seen as a word on $\mathcal{T}(S)$ (and not on S), thus $|\tau|$ is n .

49:4 Keyboards

► **Definition 8.** The length of a key t , written $|t|$, is its length as a word on S . Further, given $\sigma \in S$, we note $|t|_\sigma$ the number of occurrences of σ in t . The size of a configuration $\langle u|v \rangle$ is defined as $|\langle u|v \rangle| = |u| + |v|$.

► **Definition 9.** Two keys t and t' are equivalent, denoted $t \sim t'$, if for all $u, v \in A^*$, $\langle u|v \rangle \cdot t = \langle u|v \rangle \cdot t'$.

► **Example 10.** ε is equivalent to $a\leftarrow$ for all $a \in A$, but not to $\blacktriangleright\blacktriangleleft$, as we have $\langle a|\varepsilon \rangle \cdot \blacktriangleright\blacktriangleleft = \langle \varepsilon|a \rangle$ whereas $\langle a|\varepsilon \rangle \cdot \varepsilon = \langle a|\varepsilon \rangle$.

Example 10 illustrates how \blacktriangleleft , \blacktriangleright and \leftarrow act differently if one side of the configuration is empty. We will see that these “edge effects” add some expressiveness compared to the effective semantics, but make proofs more complex.

► **Definition 11 (Automatic Keyboard).** An automatic keyboard is a finite subset of $\mathcal{T}(S)$.

► **Definition 12.** An execution of an automatic keyboard K on a configuration $c_0 \in \mathcal{C}$ is a **non-empty** finite sequence $\rho = (t_1, c_1), \dots, (t_{n+1}, c_{n+1}) \in (K \times C)^{n+1}$ ($n \in \mathbb{N}$) such that

$$\forall i \in \llbracket 1; n+1 \rrbracket, c_{i-1} \xrightarrow{t_i} c_i.$$

By default, we take as initial configuration $c_0 = \langle \varepsilon|\varepsilon \rangle$. We usually write $c_0 \xrightarrow{\tau} c_{n+1}$ to mean the execution $(\tau[1], c_0 \cdot \tau[1]), \dots, (\tau[n+1], c_0 \cdot \tau[1, n+1])$.

► **Definition 13.** A word $w \in A^*$ is recognised by an automatic keyboard K if there exist $u, v \in A^*$ and an execution $\langle \varepsilon|\varepsilon \rangle \xrightarrow{\tau} \langle u|v \rangle$ such that $w = uv$. The language $\mathcal{L}(K)$ of K is the set of words recognised by K .

We now define keyboards as automatic keyboards to which we added some final keys “with entry”, which mark the end of the execution.

► **Definition 14 (Keyboard (with entry)).** A keyboard K on S is a pair (T, F) of finite sets $T, F \subset \mathcal{T}(S)$. We call the elements of F the final keys of K and the elements of T its transient keys.

► **Definition 15 (Accepting execution of a keyboard).** Let $K = (T, F)$ be a keyboard and $c_0 = \langle u_0|v_0 \rangle$ an initial configuration. An accepting execution of K on c_0 is a finite sequence $\rho = (t_1, c_1), \dots, (t_{n+1}, c_{n+1}) \in (T \times C)^n \cdot (F \times C)$ ($n \in \mathbb{N}$) such that

$$\forall i \in \llbracket 1; n+1 \rrbracket, c_{i-1} \xrightarrow{t_i} c_i.$$

By default, an accepting execution is on the empty configuration $\langle \varepsilon|\varepsilon \rangle$.

► **Definition 16.** A word $w \in A^*$ is recognised by a keyboard K if there exist $u, v \in A^*$ and an execution $\langle \varepsilon|\varepsilon \rangle \xrightarrow{\tau} \langle u|v \rangle$ such that $w = uv$. The language $\mathcal{L}(K)$ of K is the set of words recognised by K .

► **Example 17.** The keyboard with one transient key \mathbf{aa} and one final key \mathbf{a} , recognises sequences of a of odd length.

► **Remark 18.** Let K_a be an automatic keyboard, then $\mathcal{L}(K_a)$ is recognised by the keyboard with entry (K_a, K_a) . In all that follows, we will thus see automatic keyboards as a subclass of keyboards.

► **Definition 19** (Size of a keyboard). *The size of a keyboard $K = (T, F)$ is defined as*

$$\|K\|_\infty = \max\{|t| \mid t \in T \cup F\}.$$

We may also use another measure $|K|$ of the size of K for complexity purposes:

$$|K| = \sum_{t \in T \cup F} (|t| + 1).$$

► **Definition 20** (Minimal keyboard). *A minimal keyboard K is an automatic keyboard without any operation besides writing letters. It can therefore be seen as a finite subset of A^* . We will note MK the class of minimal keyboards.*

► **Remark 21.** We construct our keyboard classes through the sets of special operations we allow. Class names are obtained by adding B (for \leftarrow), E (for the entry, noted \blacksquare), L (for \blacktriangleleft) and A (for \blacktriangleleft and \blacktriangleright) to K. We obtain these classes.

MK : $\{\}$	LK : $\{\blacktriangleleft\}$	AK : $\{\blacktriangleleft, \blacktriangleright\}$
EK : $\{\blacksquare\}$	LEK : $\{\blacktriangleleft, \blacksquare\}$	EAK : $\{\blacktriangleleft, \blacktriangleright, \blacksquare\}$
BK : $\{\leftarrow\}$	BLK : $\{\blacktriangleleft, \leftarrow\}$	BAK : $\{\blacktriangleleft, \blacktriangleright, \leftarrow\}$
BEK : $\{\leftarrow, \blacksquare\}$	BLEK : $\{\blacktriangleleft, \leftarrow, \blacksquare\}$	BEAK : $\{\blacktriangleleft, \blacktriangleright, \leftarrow, \blacksquare\}$

► **Remark 22.** We do not consider classes with \blacktriangleright without \blacktriangleleft because, without the \blacktriangleleft operator, we can only reach configurations of the form $\langle u|\varepsilon \rangle$ and thus \blacktriangleright never has any effect.

► **Remark 23.** We use the class names above to designate both keyboard classes and language classes. For instance, we will write that L is in AK if there exists a keyboard $K \in \text{AK}$ such that $L = \mathcal{L}(K)$.

4 General properties

In this section, we establish some properties on keyboard. Although most of them are quite intuitive, we take the time to be as formal as possible in order to build solid bases for the study of keyboards.

Our first lemma states that applying a key can only affect a bounded part of the word around the cursor.

► **Lemma 24** (Locality). *Let $t = \sigma_1 \dots \sigma_n$ be a key. If $\langle u|v \rangle \xrightarrow{t} \langle u'|v' \rangle$, then $u[1, |u| - n]$ is a prefix of u' and $v[n + 1, |v|]$ is a suffix of v' .*

Furthermore, $u'[1, |u'| - n]$ is a prefix of u and $v'[n + 1, |v'|]$ is a suffix of v .

Then we formalize the fact that if the cursor is far enough from the extremities of the word, then we do not have edge effects.

► **Lemma 25** (Effectiveness far from the edges). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v \rangle$ a configuration and $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$. If $n \leq \min(|u|, |v|)$, then $\langle u|v \rangle \xrightarrow{t}_e \langle u_n|v_n \rangle$, meaning that all the arrows and backspaces are applied effectively.*

The two next lemmas bound the variation in length of the configuration when applying a key.

► **Lemma 26** (Bounds on the lengths). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v \rangle$ a configuration and $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$. Then*

$$|uv| - |t|_{\leftarrow} + \sum_{x \in A} |t|_x \leq |u_n v_n| \leq |uv| + \sum_{x \in A} |t|_x.$$

In particular $||u_n v_n| - |uv|| \leq n$. Moreover $||u_n| - |u|| \leq n$ and $||v_n| - |v|| \leq n$.

► **Lemma 27** (Length evolution without left edge effects). *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v \rangle$ a configuration such that $|u| \geq n$. Let $\langle u_n|v_n \rangle = \langle u|v \rangle \cdot t$, then*

$$|u_n v_n| = |uv| - |t|_{\leftarrow} + \sum_{x \in A} |t|_x.$$

Then, we obtain the following lemma that can be used to show that some languages are not recognised by a keyboard.

► **Lemma 28.** *Let K be a keyboard with language L . Let $(\ell_n)_{n \in \mathbb{N}}$ be the sequence obtained by sorting the lengths of the words in L by increasing order. Then $(\ell_{n+1} - \ell_n)_{n \in \mathbb{N}}$ is bounded by $3\|K\|_\infty$.*

► **Example 29.** The languages $\{a^{n^2} \mid n \in \mathbb{N}\}$ and $\{a^p \mid p \text{ prime}\}$ are not recognised by a keyboard.

The two following lemmas will be useful when studying effective executions.

► **Lemma 30.** *Let $t = \sigma_1 \dots \sigma_n$ be a key such that $\langle u|v \rangle \xrightarrow{t}_e \langle u_n|v_n \rangle$. Then, for all words x, y , $\langle xu|vy \rangle \xrightarrow{t}_e \langle xu_n|v_n y \rangle$.*

► **Lemma 31.** *Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v \rangle$ and $\langle x|y \rangle$ configurations such that $|u| = |x|$ and $|v| = |y|$. Then t acts efficiently from $\langle u|v \rangle$ if and only if it acts efficiently from $\langle x|y \rangle$.*

5 Key behaviour

This section aims at providing tools to describe the behaviour of a key. How can we formally express the intuitive fact that the i^{th} symbol of $c \cdot t$ was written by t or that the i^{th} symbol of c was moved by t ? We are going to distinguish letters from t and c in order to keep track of where t writes its letters and how the letters of c were affected.

► **Definition 32** (Tracking function). *Let \mathbb{Z}_t and \mathbb{Z}_c be two duplicates of \mathbb{Z} . We denote by \bar{k} the elements of \mathbb{Z}_c and by \hat{k} the elements of \mathbb{Z}_t .*

We define the tracking functions, one for keys $f_t: S^ \rightarrow (S \cup \mathbb{Z}_t)^*$, defined as follows: $f_t(\sigma_1 \dots \sigma_n) = \sigma'_1 \dots \sigma'_n$ where*

$$\sigma'_i = \begin{cases} \hat{i} & \text{if } \sigma_i \in A \\ \sigma_i & \text{otherwise} \end{cases}$$

and one for configurations $f_c: \mathcal{C}(A) \rightarrow \mathbb{Z}_c^ \times \mathbb{Z}_c^*$ defined by*

$$f_c(a_1 \dots a_k, b_1 \dots b_j) = \langle \bar{-k} \dots \bar{-1} \mid \bar{1} \dots \bar{j} \rangle.$$

By applying $f_t(t)$ to $f_c(c)$, we can keep track of which letters of the configuration and of the key were written, erased, or displaced, and where. We need two copies of \mathbb{Z} to differentiate between the symbols of $f_t(t)$ (added by the key) and $f_c(c)$ (already in the configuration).

► **Definition 33.** Let $\langle u|v \rangle$ be a configuration and t a key. We note $\langle u'|v' \rangle = \langle u|v \rangle \cdot t$ and $\langle x|y \rangle = f_c(u, v) \cdot f_t(t)$. We say that t writes its k^{th} symbol at position i from $\langle u|v \rangle$ if $\langle x|y \rangle_i = \widehat{k}$.

► **Remark 34.** Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v \rangle$ a configuration and $1 \leq j < k \leq n$ integers. Then t writes its k^{th} symbol at position i from $\langle u|v \rangle$ if and only if $\sigma_{j+1} \dots \sigma_n$ writes its $(k-j)^{\text{th}}$ symbol at position i from $\langle u|v \rangle \cdot \sigma_1 \dots \sigma_j$. In particular, t writes its k^{th} symbol at position i from $\langle u|v \rangle$ if and only if $\sigma_k \dots \sigma_n$ writes its 1^{st} symbol from $\langle u|v \rangle \cdot \sigma_1 \dots \sigma_{k-1}$.

We defined an intuitive notion of writing the k^{th} symbol of t . In particular, if t writes its k^{th} symbol in i^{th} position from $\langle u|v \rangle$, then $\langle u'|v' \rangle_i = t_k$, as stated below.

► **Proposition 35.** Let $t = \sigma_1 \dots \sigma_n$ be a key, $\langle u|v \rangle$ a configuration. We note

$$\begin{aligned} \langle u_n|v_n \rangle &= \langle u|v \rangle \cdot t & \langle x'_n|y'_n \rangle &= f_c(u, v) \cdot t \\ \langle u'_n|v'_n \rangle &= \langle u|v \rangle \cdot f_t(t) & \langle x_n|y_n \rangle &= f_c(u, v) \cdot f_t(t) \end{aligned}$$

Then $|u_n| = |x_n| = |u'_n| = |x'_n|$ and $|v_n| = |y_n| = |v'_n| = |y'_n|$. And for all $a \in A$,

$$\begin{aligned} \langle u_n|v_n \rangle_j = a &\text{ iff } \langle x_n|y_n \rangle_j = \bar{k} \text{ and } \langle u|v \rangle_k = a && \text{ or } \langle x_n|y_n \rangle_j = \widehat{k} \text{ and } t_k = a \\ &\text{ iff } \langle u'_n|v'_n \rangle_j = a && \text{ or } \langle u'_n|v'_n \rangle_j = \widehat{k} \text{ and } t_k = a \\ &\text{ iff } \langle x'_n|y'_n \rangle_j = \bar{k} \text{ and } \langle u|v \rangle_k = a && \text{ or } \langle x'_n|y'_n \rangle_j = a \\ &(\text{iff } a \text{ already in configuration} && \text{ or } a \text{ added by } t). \end{aligned}$$

Tracking functions are a convenient formalism to show some results on keyboards. Besides, they permit to take multiples points of view.

► **Corollary 36.** Let t be a key and $\langle u|v \rangle$ a configuration. Then t writes its k^{th} symbol at position i from $\langle u|v \rangle$ if and only if $(\langle u|v \rangle \cdot f_t(t))_i = \widehat{k}$.

► **Definition 37.** Let t be a key, $\langle u|v \rangle$ a configuration. We say that t writes an a in i^{th} position from $\langle u|v \rangle$ if there exists k such that $t_k = a$ and t writes its k^{th} symbol in position i from $\langle u|v \rangle$. We say that t writes an a from $\langle u|v \rangle$ if t writes an a in some position from $\langle u|v \rangle$.

Then, we obtain some results, which are direct consequences of Proposition 35.

► **Proposition 38.** If t writes its k^{th} symbol in i^{th} position from $\langle u|v \rangle$ then $(\langle u|v \rangle \cdot t)_i = t_k$. In particular, if t writes an a in i^{th} position from $\langle u|v \rangle$ then $(\langle u|v \rangle \cdot t)_i = a$ and if t writes an a from $\langle u|v \rangle$ then $\langle u|v \rangle \cdot t$ contains an a .

► **Proposition 39.** Let t be a key and $\langle u|v \rangle, \langle u'|v' \rangle$ two configurations such that $|u| = |u'|$ and $|v| = |v'|$. Then t writes its k^{th} symbol in i^{th} position from $\langle u|v \rangle$ if and only if t writes its k^{th} symbol in i^{th} position from $\langle u'|v' \rangle$. In particular, t writes an a in j^{th} position from $\langle u|v \rangle$ if and only if t writes an a in j^{th} position from $\langle x|y \rangle$, and t writes an a from $\langle u|v \rangle$ if and only if t writes an a from $\langle x|y \rangle$.

This proposition makes explicit the fact that keys cannot read the content of a configuration. This leads to the following characterization.

► **Proposition 40.** Let t be a key, $a \in A$ and $\langle u|v \rangle$ a configuration containing no a . Let $\langle u'|v' \rangle = \langle u|v \rangle \cdot t$. t writes an a in position i from $\langle u|v \rangle$ if and only if $\langle u'|v' \rangle_i = a$. In particular, t writes an a from $\langle u|v \rangle$ if and only if $\langle u'|v' \rangle$ contains an a .

Clearly, if the number of a 's in a configuration increases after applying a key then this key writes an a .

► **Proposition 41.** *Let t be a key and $\langle u|v \rangle$ a configuration. If $|\langle u|v \rangle|_a < |\langle u|v \rangle \cdot t|_a$, then t writes an a from $\langle u|v \rangle$.*

Note that if a key behaves differently from two configurations, then there must be some edge effects. In what follows we focus on effective executions.

► **Proposition 42.** *Let t be a key and $\langle u|v \rangle$ and $\langle x|y \rangle$ two configurations on which t acts effectively. Then t writes its k^{th} symbol in i^{th} position from $\langle u|v \rangle$ if and only if t writes its k^{th} symbol in i^{th} position from $\langle x|y \rangle$. Therefore, t writes an a in position i from $\langle u|v \rangle$ if and only if t writes an a in position i from $\langle x|y \rangle$.*

In other words, a key always behaves the same way far from the edges of the configuration.

► **Definition 43.** *Let t be a key. We say that t ensures an a in position i far from the edges if there exists a configuration $\langle u|v \rangle$ such that t acts effectively on $\langle u|v \rangle$ and t writes an a in position i from $\langle u|v \rangle$.*

Then, we immediately obtain the following propositions.

► **Proposition 44.** *Let t be a key and $u, v \in A^*$ such that $|u| \geq |t|$ and $|v| > |t|$. Then t ensures an a far from the edges if and only if t writes an a from $\langle u|v \rangle$.*

► **Proposition 45.** *Let t be a key and $u, v \in A^*$ such that $|u| \geq |t|$ and $|v| \geq |t|$. If $|\langle u|v \rangle|_a < |\langle u|v \rangle \cdot t|_a$, then t ensures an a far from the edges.*

► **Proposition 46.** *Let t be a key which ensures an a far from the edges and $\langle u|v \rangle$ such that $|u| \geq |t|$ and $|v| \geq |t|$. Then $\langle u|v \rangle \cdot t$ contains an a .*

6 Characterisation of the classes

6.1 Languages of BEK (without the arrows)

To begin, we study keyboards that do not contain any arrows. As these keyboards has no \blacktriangleleft operation, the right component of a configuration in an execution of $K \in \text{BEK}$ (starting from $\langle \varepsilon|\varepsilon \rangle$) is always empty. Thus, in this part, we will sometimes denote u for the configuration $\langle u|\varepsilon \rangle$.

6.1.1 MK and EK

MK and EK are quite easy to understand. Indeed, since a key of a minimal keyboard K is just a word on A , $K \subset A^*$.

► **Remark 47.** Let $K = \{w_1, \dots, w_n\}$ be a minimal keyboard. Then $\mathcal{L}(K) = (w_1 + \dots + w_n)^+$.

EK languages are rather similar.

► **Lemma 48.** *Let $K = (T, F)$ be a EK keyboard. Then, $\mathcal{L}(K) = T^*F$ and this regular expression can be computed in $O(|K|)$.*

Thus, we can build in linear time a regular expression that recognises $\mathcal{L}(K)$.

6.1.2 BK and BEK

Some of the expressiveness of BEK comes from edge effects. For instance, finite languages are recognised by BK keyboards.

► **Example 49.** Let L be a finite language and $M = \max\{|w| \mid w \in L\}$. Then L is recognised by $K = \{\leftarrow^M w \mid w \in L\}$.

These edge effects could make BEK languages quite complex. On the other hand, we show that BEK keys can be put in a particular form.

► **Lemma 50 (Normal form).** *Let $t \in S^*$ be a key from BEK. Then there exist $m \in \mathbb{N}$ and $w \in A^*$ such that $t \sim \leftarrow^m w$. Further, m and w can be computed from t in polynomial time.*

Using this normal form, we understand that the action of a BEK key always consists in deleting a bounded number of letters at the end of the word, then adding a bounded number of letters. This reminds us of stacks. Following this intuition, we can easily encode the behaviour of a BEK keyboard into a pushdown automaton.

However, BEK is even more narrow since all languages of BEK are regular.

► **Theorem 51.** *Let K be a BEK keyboard. Then, $\mathcal{L}(K)$ is regular, and we can build an NFA $\mathcal{A}(K)$ recognising $\mathcal{L}(K)$ in polynomial time.*

► **Remark 52.** There are several ways to prove this result. One of them is to apply the pushdown automaton construction from the proof of Theorem 57 (presented later in the paper) in the particular case of BEK. The language of the BEK keyboard is then essentially the stack language of this automaton. A slight adaptation of the classical proof that the stack language of a pushdown automaton is rational then yields the result.

We choose to include another proof in this work, as it is elementary, not much longer than the one aforementioned, and seems more elegant to the authors.

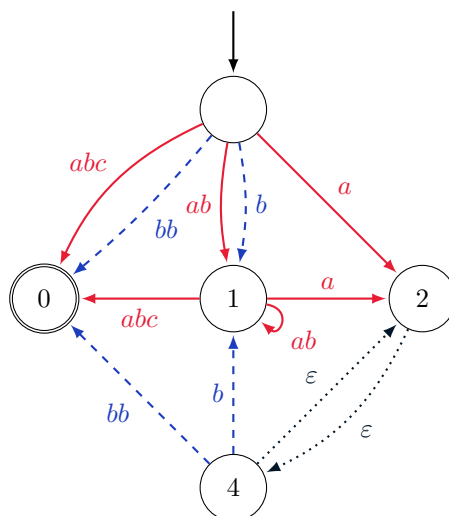
We first show the result for languages of BK. Let K be an BK keyboard. The key idea is that at any point of the execution, we can split the word into:

- a prefix v of letters that will never be erased,
- a suffix x of letters which will eventually be erased.

As an example, in order to write abb with the keys $\leftarrow^2 abcc$, $\leftarrow^3 b$ and $\leftarrow^2 cbc$, we can write ab with t_1 and the remaining b with t_2 . However, when applying t_1 and then t_2 , t_2 erases too many letters and we obtain ab instead of abb . We therefore need a sequence of keys which does not erase the prefix ab , but replaces the two c by three letters, allowing us to apply t_2 . This is done by applying t_3 , replacing cc with cbc . We can therefore write abb with the sequence $t_1 t_3 t_2$.

Given a keyboard K of BK, we construct an automaton following this idea. Its states are integers from 0 to $\|K\|_\infty$, state i meaning that we have i extra letters to be erased at the end of the current word.

- A transition from i to j labelled by $w \in A^+$ means there exists a key erasing i letters and writing wx with $|x| = j$ (w is a part of the word we are trying to recognise and x is a suffix of extra letters to be erased).
- A transition from i to j labelled by ε means there exists a sequence of keys turning the suffix of extra letters of length i into one of length j , without erasing any of the other letters.



■ **Figure 1** Automaton corresponding to keyboard $\{\leftarrow\text{abc}, \leftarrow^4\text{bb}\}$.

Figure 1 shows the automaton obtained from keyboard $\{\leftarrow\text{abc}, \leftarrow^4\text{bb}\}$ with this construction (slightly simplified as we erased some useless states).

The red transitions (full arrows) are the ones we can do with $\leftarrow\text{abc}$ and the blue (dashed) ones those we can do by applying $\leftarrow^4\text{bb}$. The black (dotted) transitions, are the ones we use to switch from i to j extra letters.

As an example, let us see how we can recognise *babababb* with this automaton and this keyboard. We decompose it into $b \cdot a \cdot b \cdot ab \cdot abc$.

- We write a b with $\leftarrow^4\text{bb}$, leading us to state 1 (we need to erase the second b before to write the rest of the word).
- The key $\leftarrow\text{abc}$ then allows us to erase the second b and write a , leading to state 2 (we need to erase the bc suffix).
- A sequence of keys then allows us to switch from two extra letters (state 2 in the automaton) to four (state 4).
- The key $\leftarrow^4\text{bb}$ then allows us to erase four letters and write b , leaving us in state 1.
- The key $\leftarrow\text{abc}$ erases b and writes ab , leaving us in state 1.
- The key $\leftarrow\text{abc}$ erases the last letter and writes abc , with no extra letters, leaving us in state 0 in the automaton, where we can accept.

We need to construct the transitions of our NFA recognising the language of an BK keyboard. The construction of the transitions labelled by letters is rather straightforward, whereas the construction of the ε -transitions requires some arithmetic arguments, which are detailed in the full version of this paper.

We slightly adapt this construction to obtain an automaton for a keyboard of BEK.

- The state 0 is no longer final, and we add a final state Fin .
- For all states i and $t_f = \leftarrow^i w \in F$, we add a transition from i to Fin labelled by w . This simulates the action of, after producing vx (with $|x| = i$) with T , applying t_f .
- For all $t_f = \leftarrow^r w \in F$, we add a transition from Init to Fin labelled by w .

6.2 Languages of BLEK (without the right arrow)

In this section, we allow the use of \blacktriangleleft . With this symbol, we have the possibility to move into the word, and then erase or write letters. It opens a new complexity level.

Moreover, we provide a non-regular language of BLEK, hence showing that it is more expressive than BEK (see Theorem 51).

► **Example 53.** Let $K = \{aa\blacktriangleleft, bb\blacktriangleleft\}$. Then, $\mathcal{L}(K) = \{u\tilde{u} \mid u \in (a+b)^+\}$, that is, K recognises the non-empty palindromes of even length.

Thus, we can represent context-free non-regular languages with BLEK (one can observe that the keyboard of Example 53 is actually even in LK).

However, a basic observation helps us to understand the behaviour of a key of BLEK: as we do not have the symbol \blacktriangleright , we cannot go back to the right and all the letters to the right of the cursor are written forever. The following lemma can be proven easily by induction.

► **Lemma 54.** *Let $t = \sigma_1 \dots \sigma_n$ be a sequence of atomic operations, and $\langle u|v \rangle$ a configuration. Then, $\langle u|v \rangle \cdot t$ is of the form $\langle u'|v' \rangle$.*

Then, we can make some assertions about a key observing its result over a configuration.

► **Lemma 55 (Independence from position).** *Let t be a key of BLEK and $\langle u|v \rangle$ a configuration. If t writes an a from $\langle u|v \rangle$, then for all configurations $\langle u'|v' \rangle$, t writes an a from $\langle u'|v' \rangle$.*

Proof. We set $t = \sigma_1 \dots \sigma_n$. Let $\langle x|y \rangle$ be a configuration. By Proposition 39, we can assume it does not contain any a and by Remark 34, we can assume $\sigma_1 = a$ and t writes its first symbol from $\langle u|v \rangle$. We then define for all $1 \leq i \leq n$

$$\langle u_i|v_i \rangle = \langle u|v \rangle \cdot \sigma_1 \dots \sigma_i \quad \text{and} \quad \langle x_i|y_i \rangle = \langle x|y \rangle \cdot \sigma_1 \dots \sigma_i.$$

Let i be the smallest index such that u_i or x_i is empty, and $i = n$ if such an index does not exist (note that $i > 1$ as u_1 and x_1 contain the a written by σ_1). As t writes its first symbol from $\langle u|v \rangle$, $\sigma_1 \dots \sigma_i$ writes an a from $\langle u|v \rangle$ in some position j .

Further, $\sigma_1 \dots \sigma_i$ acts efficiently on $\langle u|v \rangle$ and on $\langle x|y \rangle$, thus by Proposition 42, t writes an a from $\langle x|y \rangle$ at position j . If $i = n$, the result is proven.

Otherwise, as either u_i or x_i is empty, we have $j > 0$ and $y_i[j] = v_i[j] = a$. By Lemma 54, y_n contains an a , while $\langle x|y \rangle$ does not by assumption, showing the result by Proposition 40. \blacktriangleleft

Moreover, we can refine Lemma 54.

► **Theorem 56 (BLEK fundamental).** *Let $t = \sigma_1 \dots \sigma_n$ be a sequence of atomic operations, and $\langle u|v \rangle$ a configuration. We set $\langle x_n|y_n \rangle = \langle \varepsilon|\varepsilon \rangle \cdot t$. Then $\langle u|v \rangle \cdot t$ is of the form $\langle u_n x_n|v_n v \rangle$ with y_n a subword of v_n and u_n a prefix of u .*

These observations help us to better understand BLEK. A key observation is that we can see the left part of the configuration as a stack, which can be modified, and the right part as the fixed one, just as the prefix of a word that has been read by an automaton. We can then recognise (the mirror of) a BLEK language with a pushdown automaton which guesses a sequence of keys, maintains the left part of the configuration in the stack and reads the right part of the configuration.

► **Theorem 57.** *Let K be a BLEK keyboard. Then, $\mathcal{L}(K)$ is context-free, and we can build a non-deterministic pushdown automaton $\mathcal{A}(K)$ recognising $\mathcal{L}(K)$ in polynomial time.*

The idea is to use the fact that what is written to the right of the cursor is unaffected by a key without right arrows (see Theorem 56).

We construct a pushdown automaton maintaining the invariant “After simulating the application of keys t_1, \dots, t_n we have read a word v and have as stack content a word u such that $\langle \varepsilon | \varepsilon \rangle \cdot t_1 \cdots t_n = \langle u | \tilde{v} \rangle$ ”.

Atomic operations are easily translated to satisfy that invariant. Writing a letter comes down to pushing it on the stack, applying a backspace to deleting the stack head, and applying a left arrow to popping and reading the stack head.

After simulating a sequence of keys, the automaton pops and reads the content of its stack before to accept. At this point, the automaton has read the mirror of the word produced by this sequence of keys. We then simply construct a pushdown automaton recognising the mirror of the language of the previous one.

6.3 Languages of EAK (without backspace)

A third interesting class is EAK, where the backspace is not allowed. Thus, the size of a configuration does not decrease along an execution. The execution of such a keyboard can therefore be easily simulated on a linear bounded automaton, as stated in Theorem 60.

In all the proofs of this section we will say that t writes an a when t contains an a (as we have no \leftarrow if t contains an a then this a will not be erased when applying t).

► **Lemma 58.** *Let $K = (T, F)$ be a EAK keyboard. Let $u, v \in A^*$, let $\tau \in (T \cup F)^*$ and let $\langle u' | v' \rangle = \langle u | v \rangle \cdot \tau$. Then uv is a subword of $u'v'$. In particular $|uv| \leq |u'v'|$.*

► **Lemma 59.** *Let $K = (T, F)$ be a EAK keyboard, let $w \in \mathcal{L}(K)$. There exists an execution $\tau = t_1 \cdots t_n \in T^*F$ such that $\langle \varepsilon | \varepsilon \rangle \cdot \tau = \langle u | v \rangle$ with $uv = w$ and $n \leq |w|^2 + 1$.*

Proof. Let $w \in \mathcal{L}(K)$, and let $u, v \in A^*$, $\tau = t_1 \cdots t_n \in T^*F$ be such that $w = uv$ and $\langle \varepsilon | \varepsilon \rangle \cdot \tau = \langle u | v \rangle$.

We show that if $n > |w|^2 + 1$ then there exists a shorter execution writing $\langle u | v \rangle$.

Suppose $n > |w|^2 + 1$, and for all $0 \leq i \leq n$ let $k_i = |\langle \varepsilon | \varepsilon \rangle \cdot t_1 \cdots t_i|$ and $\langle u_i | v_i \rangle = \langle \varepsilon | \varepsilon \rangle \cdot t_1 \cdots t_i$. By Lemma 58, the sequence (k_i) is nondecreasing. As $n > |w|^2 + 1$ and $k_n = |w|$, there exists $0 \leq i \leq n - |w|$ such that $k_i = k_{i+1} = \cdots = k_{i+|w|}$. Again by Lemma 58, we have $u_i v_i = u_{i+1} v_{i+1} = \cdots = u_{i+|w|} v_{i+|w|}$. If $u_i v_i = w$ then the execution $t_1 \cdots t_i$ writes w .

If $u_i v_i \neq w$ then $k_i < |w|$. There are $|w| > k_i$ configurations $\langle u' | v' \rangle$ such that $u'v' = u_i v_i$, thus there exist $i \leq j_1 < j_2 \leq i + |w|$ such that $\langle u_{j_1} | v_{j_1} \rangle = \langle u_{j_2} | v_{j_2} \rangle$.

As a result, the execution $t_1 \cdots t_{j_1} t_{j_2+1} \cdots t_n$ writes w .

The lemma is proven. ◀

► **Theorem 60.** *For all keyboards $K = (T, F)$ of EAK we can construct a linear bounded automaton $\mathcal{A}(K)$ of polynomial size recognising $\mathcal{L}(K)$.*

Proof. We construct $\mathcal{A}(K)$ the linear bounded automaton which, given an input w , proceeds as follows:

- It divides the tape in three parts: one to memorize the input (of linear size), one to simulate an execution of K (of linear size as well by Lemma 58) and one containing a counter (of logarithmic size).
- It guesses a sequence of keys of T followed by a key of F and computes their effect on the fly. After the application of each key, the counter is incremented.

- If the counter goes beyond $|w|^2 + 1$, then the automaton rejects.
- If not, the automaton compares the obtained word to w , accepts if they are equal, and rejects otherwise.

This machine guesses a sequence of at most $|w|^2 + 1$ keys and accepts if the word obtained by their actions is the input.

Clearly if a word is accepted by $\mathcal{A}(K)$ then it is in $\mathcal{L}(K)$. Conversely, if a word w is in $\mathcal{L}(K)$ then by Lemma 59 there exists an execution of length at most $|w|^2 + 1$ accepting it, thus $\mathcal{A}(K)$ can guess this execution and accept w .

As a result, $\mathcal{L}(K) = \mathcal{L}(\mathcal{A}(K))$. ◀

7 Comparison of the keyboard classes

The characterisations that we provide give us some information about each class independently. We now compare the subclasses of BEAK in order to find out which inclusions hold between them. One of these inclusions, between BAK and BEAK, is especially interesting since it shows that keyboards with entry are strictly more powerful than automatic ones.

We decompose our results into the following propositions. Those establish that a class is included in another if and only if that same inclusion holds between their sets of operators, except possibly for the inclusion of EK and BEK in BAK, which we do not prove or disprove.

To start with, we show that a class containing the left arrow cannot be included in a class lacking it. This is a direct consequence of Example 53 and Theorem 51 as we have a language of LK which is not rational, and thus not in BEK.

► **Proposition 61.** $LK \not\subseteq BEK$.

We continue with the two next propositions, showing that a class containing the entry cannot be included in a class excluding it, except possibly for BAK.

► **Proposition 62** ($EK \not\subseteq BLK$). *EK is not included in BLK.*

► **Proposition 63** ($EK \not\subseteq AK$). *EK is not included in AK.*

Then we prove that a class with \leftarrow cannot be included in a class without \leftarrow .

► **Proposition 64** ($BK \not\subseteq EAK$). *BK is not included in EAK.*

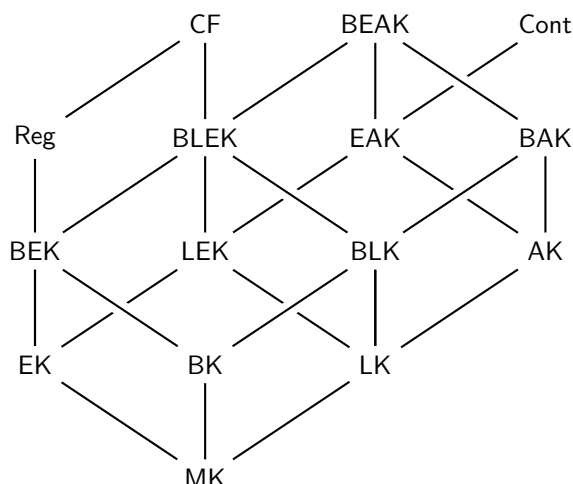
The next proposition states that a class containing \blacktriangleright cannot be included in a class lacking it.

► **Proposition 65** ($AK \not\subseteq BLEK$). *AK is not included in BLEK.*

And finally we show that, except possibly for EK and BEK, a class with entry cannot be included in a class without entry.

► **Proposition 66** ($LEK \not\subseteq BAK$). *LEK is not included in BAK.*

The inclusion Hasse diagram of all subclasses of BEAK and traditional language classes is displayed in Figure 2.



■ **Figure 2** Hierarchy of language classes.

8 Complexity results

In this section we establish some complexity upper bounds on the membership and universality problems for various keyboard classes. The following three propositions are direct consequences of the known complexity bounds of the models which we translated keyboards into (in Remark 47, Lemma 48, Theorem 51 and Theorem 57).

► **Proposition 67.** *The membership problem on MK and EK is in PTIME. The universality problem is in PTIME on MK and PSPACE on EK.*

The problem for EK seems simple: it amounts to deciding, given two finite sets of words T and F , if T^*F is universal.

► **Proposition 68.** *The membership problem over BEK is in PTIME, and the universality problem over BEK in PSPACE.*

► **Proposition 69.** *The membership problem over BLEK is in PTIME.*

For BK keyboards, we prove that there exists a word not accepted by a given BK keyboard if and only if there exists one of polynomial length.

► **Proposition 70.** *The universality problem for BK keyboards is in CONP.*

For EAK keyboards, we know by Lemma 59 that every word w recognised by a EAK keyboard can be written with an execution of length polynomial in $|w|$, hence this proposition:

► **Proposition 71.** *The membership problem for EAK keyboards is in NP.*

9 Closure properties

In this section, we study closure properties of keyboard classes. We selected three operators, the union and the intersection, as they are the most natural closure operators, and the mirror, under which some classes are stable.

► **Proposition 72 (Mirror).** *MK, AK and EAK are stable by mirror. EK, BK, BEK and BLK are not stable by mirror.*

► **Proposition 73** (Intersection). *None of the keyboard language classes are stable by intersection.*

► **Proposition 74** (Union). *None of the keyboard language classes are stable by union.*

We end this section with an undecidability result, showing that intersecting keyboards can lead to highly complex languages. This shows another link with context-free languages, as the emptiness of the intersection of two context-free languages is undecidable as well.

► **Proposition 75** (Intersection emptiness problem). *The following problem is undecidable:*

Input: K_1, K_2 two LK keyboards.

Output: Is $\mathcal{L}(K_1) \cap \mathcal{L}(K_2)$ empty?

10 Conclusion

A natural question when it comes to models of computation is what we can do without any memory or any information on the current state of the system. We initiated a line of research aiming at studying such “blind” models. The one we considered here, keyboards, proved to be mathematically complex and interestingly orthogonal in expressiveness to several of the most classical models. We have established a number of properties of keyboards, as well as a vocabulary facilitating their study. We separated almost all classes and compared their expressiveness, thereby uncovering the lattice of their power.

Future work

As keyboards are a completely new model, there are many open problems we are working on or intend to address. We conjecture that EK is not included in BAK, but we do not have a proof. We also conjecture that not all AK languages are algebraic (the language generated by $\{a\blacktriangleright\blacktriangleright, b\blacktriangleleft\blacktriangleleft\}$ is a candidate as a counter-example) and that BEAK does not contain all rational languages ($a^* + b^*$ being a potential counter-example). We plan on extending the set of operations to add, for instance, a right erasing operator, symmetric to \leftarrow . It could also be interesting to study the semantics in which we forbid non-effective operations. Finally, we could equip the model with states and transitions labelled with keys. We would then have more control over which keys are applied at which times, thus increasing the expressiveness of the model and facilitating its study.

References

- 1 L. Babai and E. Szemerédi. On the complexity of matrix group problems i. In *25th Annual Symposium on Foundations of Computer Science, 1984.*, pages 229–240, 1984. doi:10.1109/SFCS.1984.715919.
- 2 Martin Beaudry. Membership testing in commutative transformation semigroups. *Information and Computation*, 79(1):84–93, 1988. doi:10.1016/0890-5401(88)90018-1.
- 3 Jean Berstel, Dominique Perrin, and Christophe Reutenauer. *Codes and Automata*, volume 129 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2010.
- 4 Petr Jančar, František Mráz, and Martin Plátek. Forgetting automata and the Chomsky hierarchy. In *Proc. SOFSEM'92*, 1993.
- 5 Petr Jančar, František Mráz, and Martin Plátek. Characterization of context-free languages by erasing automata. In Ivan M. Havel and Václav Koubek, editors, *Mathematical Foundations of Computer Science 1992*, pages 307–314, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

- 6 Rajeev Motwani John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson Education, Limited, 2013.
- 7 Neil D. Jones and William T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3(1):105–117, 1976. doi:10.1016/0304-3975(76)90068-2.
- 8 Michael S Paterson. Unsolvability in 3×3 matrices. *Studies in Applied Mathematics*, 49(1):105–107, 1970.
- 9 Burchard von Braunmühl and Rutger Verbeek. Finite-change automata. In K. Weihrauch, editor, *Theoretical Computer Science 4th GI Conference*, pages 91–100, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg.

A Comparisons between classes

► **Proposition 62** (EK $\not\subseteq$ BLK). EK is not included in BLK.

Proof. The language $L = (a^2)^*(b + b^2)$ is clearly in EK. We first show that a keyboard K of BLK recognising L does not use the \blacktriangleleft operation. The idea is that if we could reach a configuration with letters to the right of the cursor, we could then apply a sequence of keys writing b^2 from $\langle \varepsilon | \varepsilon \rangle$, which would yield a word with letters after the second b (thus not in L) by Theorem 56.

K is therefore an BK keyboard which can write b^2 , hence there exists a sequence of keys of K whose normal form is $\leftarrow^k b^2$ for some k . We reach a contradiction by considering a word $w \in L$, applying τ , and distinguishing cases according to the parity of k . ◀

► **Proposition 63** (EK $\not\subseteq$ AK). EK is not included in AK.

Proof. Consider the language $\{a\}$. It is recognised by the keyboard $(\emptyset, \{a\})$ of EK.

Suppose there exists K a keyboard of AK recognising $\{a\}$. Then there exists $t \in K$ exactly containing an a . As K does not contain any \leftarrow , applying t twice from $\langle \varepsilon | \varepsilon \rangle$ yields a configuration with two a , and thus a word outside of $\{a\}$. We obtain a contradiction. ◀

Proof of Proposition 64

Define $t_a = \leftarrow a \diamond \blacklozenge$ and $t_b = \leftarrow \leftarrow b \diamond \blacklozenge \blacklozenge$, and $L_{\diamond \blacklozenge}$ as the language recognised by the keyboard $\{t_a, t_b\}$. $L_{\diamond \blacklozenge}$ is in BK, and we show that it is not in EAK using mainly the non-erasing property of EAK keyboards.

In all that follows, we will use the notations $t_a = \leftarrow a \diamond \blacklozenge$ and $t_b = \leftarrow \leftarrow b \diamond \blacklozenge \blacklozenge$.

We define $L_{\diamond \blacklozenge}$ as the language recognised by the BK keyboard $\{t_a, t_b\}$.

► **Lemma 76.** Let $x = x_1 \cdots x_n \in \{a, b\}^+$, we have

$$\langle \varepsilon | \varepsilon \rangle \cdot t_{x_1} \cdots t_{x_n} = x_1 w_{x_1 x_2} x_2 w_{x_2 x_3} x_3 \cdots w_{x_{n-1} x_n} x_n v_{x_n}$$

with $w_{aa} = w_{bb} = \diamond$, $w_{ab} = \varepsilon$, $w_{ba} = \diamond \blacklozenge$, $v_a = \diamond \blacklozenge$ and $v_b = \diamond \blacklozenge \blacklozenge$.

In particular, for all $u_1, u_2, u_3 \in A^*$, if $u_1 b u_2 a u_3 \in L_{\diamond \blacklozenge}$ then u_2 contains a \blacklozenge .

► **Proposition 64** (BK $\not\subseteq$ EAK). BK is not included in EAK.

Proof. Suppose there exists a keyboard $K = (T, F)$ of EAK recognising $L = L_{\diamond \blacklozenge}$. In this proof, we will use the letter k to denote keys of K , in order to avoid confusion with the keys t_a and t_b .

As $a \diamond \blacklozenge \in L$, there exists $\tau_f \in T^* F$ yielding that word.

Moreover, $w = (a\Diamond)^{3\|K\|_\infty} a\Diamond \in L$ (obtained by applying t_a $3\|K\|_\infty$ times). Hence, there exists an execution of K writing that word. By Lemma 58, as $|k|_a \leq \|K\|_\infty$ for all $k \in T \cup F$, this execution contains at least three keys containing an a , including at least two in T (as the execution only has one final key).

As w contains a single \Diamond , there exists a key k_a of T writing an a , but writing neither \Diamond nor b . By applying k_a then τ_f , we obtain a word of L containing a single \Diamond and no b . By lemma Lemma 76 this word must be of the form $(a\Diamond)^i a\Diamond$. We infer that k_a contains as many a and \Diamond . Let n be its number of a .

We can similarly show, using the word $(b\Diamond)^{3\|K\|_\infty} b\Diamond$, that there exists $k_b \in T$ containing as many b and \Diamond but neither \Diamond nor a . Let m be its number of b (and of \Diamond).

Let $\tau = k_a k_b$. We write

$$\langle u|v \rangle = \langle \varepsilon|\varepsilon \rangle \cdot \tau \quad \text{and} \quad \langle u'|v' \rangle = \langle u|v \rangle \cdot \tau_f$$

We have that $u'v'$ contains a single \Diamond . As $u'v' \in L$, $u'v'$ has to be of the form $wa\Diamond$ with w not containing any \Diamond , $|w|_a = n$, $|w|_b = m$ and $|w|_\Diamond = n + m$.

As wa contains at least a b , wa is of the form $u_1 b u_2 a$, thus by Lemma 76, u_2 contains a \Diamond , yielding a contradiction. As a conclusion, $L \notin \text{EAK}$. \blacktriangleleft

Proof of Proposition 65

Consider the keyboard $K = \{\mathbf{a} \blacktriangleleft^2 \blacktriangleright \mathbf{b}\}$.

By applying $\mathbf{a} \blacktriangleleft^2 \blacktriangleright \mathbf{b}$ on $\langle \varepsilon|\varepsilon \rangle$ we obtain $\langle ab|\varepsilon \rangle$, and by applying it on a configuration of the form $\langle ub|v \rangle$ we obtain $\langle ubb|av \rangle$. Hence, after applying it n times on $\langle \varepsilon|\varepsilon \rangle$ we get $ab^{n+1}a^n$. The language of K is therefore $L = \{ab^{n+1}a^n \mid n \in \mathbb{N}\}$.

► **Lemma 77.** *Let K be a keyboard of BLEK. If K recognises L then, for all $\tau \in T^*$, $\langle \varepsilon|\varepsilon \rangle \cdot \tau$ is of the form $\langle u|a^k \rangle$.*

Proof. As $abba \in L$, there exists $\tau \in T^*$ and $t_f \in F$ such that $\langle \varepsilon|\varepsilon \rangle \cdot \tau t_f = \langle x|y \rangle$ with $xy = abba$.

Let $\langle u|v \rangle$ be a configuration reachable by a sequence of keys of τ with v containing a b . By Theorem 56, $\langle u|v \rangle \cdot \tau t_f$ is of the form $\langle u'x|v'v \rangle$ with y a subword of v' . As a consequence, $abba$ is a subword of $u'xv'v$. With the assumption, we get that $abbab$ is a subword of $u'xv'v$, which contradicts the fact that $u'xv'v \in L$. \blacktriangleleft

► **Proposition 65** (AK $\not\subseteq$ BLEK). *AK is not included in BLEK.*

Proof. Let $K = (T, F)$ be a keyboard of BLEK, suppose it recognises L . Let $\tau \in T^*$ and $t_f \in F$. We set:

$$\langle x|y \rangle = \langle \varepsilon|\varepsilon \rangle \cdot t_f \quad \text{and} \quad \langle u|v \rangle = \langle \varepsilon|\varepsilon \rangle \cdot \tau.$$

There exists $n \in \mathbb{N}$ such that $xy = ab^{n+1}a^n$. By Theorem 56, $\langle \varepsilon|\varepsilon \rangle \cdot \tau t_f = \langle u|v \rangle \cdot t_f$ is of the form $\langle u'x|v'v \rangle$ with y a subword of v' .

As $ab^{n+1}a^n$ is a subword of xv' and $u'xv'v \in L$, u' is necessarily empty (otherwise u' would contain an a and as ab is a subword of xv' , aab would be a subword of $u'xv'v$, contradicting $u'xv'v \in L$).

By Lemma 26, $|v'v| - |v| \leq |t| \leq \|K\|_\infty$, i.e., $|v'| \leq \|K\|_\infty$. Furthermore, again by Lemma 26, $|x| \leq \|K\|_\infty$.

By Lemma 77, v is of the form a^k . Hence, all b in $u'xv'v$ are in xv' , thus $u'xv'v$ contains at most $2\|K\|_\infty$ b .

We have shown that all words in $\mathcal{L}(K)$ contain at most $2\|K\|_\infty$ b , contradicting $\mathcal{L}(K) = L$ as the number of b of words of L is unbounded. \blacktriangleleft

Proof of Proposition 66

Consider the following language over $A = \{a, b, c\}$:

$$L = L_1 \cup L_2 \text{ with } L_1 = \{wc\tilde{w} \mid w \in \{a, b\}^*\} \text{ and } L_2 = \{wcc\tilde{w} \mid w \in \{a, b\}^*\}.$$

We are going to prove that L is in LEK but not in BAK. L is recognised by $K = (\{aa\blacktriangleleft, bb\blacktriangleleft\}, \{c, cc\})$, then L is in LEK.

We now show that $L \notin \text{BAK}$. The intuition is as follows:

Suppose we have a keyboard K of BAK recognising L . As we want to recognise palindromes, we need to stay close to the centre in order to always modify both halves of the word.

If the word is large enough, this means that we have to get far from the edges. In particular, there is a key t writing an a far from the edges (even two a , as we have to stay in the language).

We study the behaviour of this key on $b^n cb^n$ and $b^n ccb^n$ with large n . The cursor being far from the edges, t behaves the same way in both cases. In particular, we make the following remarks:

- The maximal distance d between two a will be the same in both words. Thus, the resulting words have either both two c or both one c in the centre.
- In both cases the key adds a number of letters δ , and thus if $b^n cb^n$ is turned into a word of even length, then $b^n ccb^n$ is turned into a word of odd length, and vice-versa. As a result, they have different numbers of c in the centre.

As those facts are contradictory, we conclude that there cannot exist such a keyboard.

► **Proposition 66** (LEK $\not\subseteq$ BAK). LEK is not included in BAK.

B Complexity

Proof of Proposition 70

► **Lemma 78.** Let K be an BK keyboard. For all $\tau \in K^*$ there exists $\tau' \in K^*$ whose normal form is $\leftarrow^{k'} w'$ with $k' \leq \|K\|_\infty$ such that $\varepsilon \cdot \tau = \varepsilon \cdot \tau'$.

Proof. We proceed by induction on τ . The result hold for $\tau = \varepsilon$ (by taking $\tau' = \varepsilon$). Let $\tau \in K^*$, suppose there exists $\tau' \in K^*$ whose normal form is $\leftarrow^{k'} w'$ with $k' \leq \|K\|_\infty$ such that $\varepsilon \cdot \tau = \varepsilon \cdot \tau'$. Let $t \in K$, we study the sequence $t\tau$. We need to provide τ'' such that $\varepsilon \cdot t\tau = \varepsilon \cdot \tau''$ and with normal form $\leftarrow^{k''} w''$ with $k'' \leq \|K\|_\infty$.

There exist k and w such that $\leftarrow^k w$ is the normal form of t . Let $n = |w| - k'$.

- If $n > 0$, then $t\tau'$ is equivalent to $\leftarrow^k w[1, n]w'$ and we can set $\tau'' = t\tau'$ (as $n > 0$, we have $k' < |w| < \|K\|_\infty$).
- Otherwise, $\varepsilon \cdot t\tau = w \cdot \tau' = w \cdot \tau' = w' = \varepsilon \cdot \tau'$ and we can set $\tau'' = \tau'$.

The result is proven. ◀

► **Proposition 70.** The universality problem for BK keyboards is in CONP.

Proof. We prove that if a BK keyboard K is not universal, then there is a word of length at most $\|K\|_\infty + 1$ it does not recognise.

Let K be a keyboard of BK, suppose there exists $w \in A^*$ not recognised by K . We take w of minimal length. If $|w| \leq \|K\|_\infty + 1$ then the property holds.

If $|w| > \|K\|_\infty + 1$ then there exist $a \in A, v \in A^+$ such that $av = w$. As we assumed w to be of minimal length, v is recognised by K . By Lemma 78, there exist $k \leq \|K\|_\infty, \tau \in K^*$ whose normal form is $\leftarrow^k v$, such that $\varepsilon \cdot \tau = v$.

As $|a^{k+1}| = k + 1 \leq \|K\|_\infty + 1 < |w|$, a^{k+1} is recognised by K . Let τ' be such that $\varepsilon \cdot \tau' = a^{k+1}$, then we have $\varepsilon \xrightarrow{\tau'} a^{k+1} \xrightarrow{\tau} av = w$.

This contradicts the fact that w is not recognised by K . The property is proven. ◀

C Closure properties

Proof of Proposition 72

► **Proposition 72** (Mirror). *MK, AK and EAK are stable by mirror. EK, BK, BEK and BLK are not stable by mirror.*

For MK the result is clear as the mirror of $\mathcal{L}(K)$ is $\mathcal{L}(\tilde{K})$ when K is an MK keyboard.

As for AK and EAK, we first define the mirror of an atomic operation: the mirror of ◀ is ▶ (and vice-versa) and the mirror of a is $a\blacktriangleleft$. By turning every atomic operation in a keyboard into its mirror, we obtain a keyboard recognising the mirror language.

The language $L = b^*a$ is in EK and BK but its mirror is not in BEK, thus EK, BK and BEK are not stable under mirror.

The language $L = (b + b^2)a^*$ is recognised by the BLK keyboard

$$K = \{\leftarrow^2 a\blacktriangleleft b, \leftarrow^2 a\blacktriangleleft bb, \leftarrow^2 b, \leftarrow^2 bb\}$$

but its mirror $a^*(b + b^2)$ is not in BLK as is shown in the proof of Proposition 62.

Proof of Proposition 73

► **Proposition 73** (Intersection). *None of the keyboard language classes are stable by intersection.*

For MK and EK, we use the intersection of $(ab + ba + bb)^*$ and $(ba + b)^*$ as our counterexample.

For BK and BEK, we use again the language $L_{\diamond\blacklozenge}$ from the proof of Proposition 64. We also define $L' = (a + b + \diamond)^+\blacklozenge^2$ recognised by the BK keyboard

$$\{\leftarrow^2 a\blacklozenge^2, \leftarrow^2 b\blacklozenge^2, \leftarrow^2 \diamond\blacklozenge^2\}.$$

Thus L and L' are in BK, and we show that their intersection is not.

For the other classes, we show that $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ is not in BEAK. Further, it is the intersection of the following LK languages:

- $(a + b)^* c^*$, recognised by $\{a, b, c\blacktriangleleft\}$,
- $a^*(b + c)^*$, recognised by $\{a, b\blacktriangleleft, c\blacktriangleleft\}$,
- $\{w \in (a + b + c)^* \mid |w|_a = |w|_b\}$, recognised by $\{ab, ba, c, \blacktriangleleft\}$,
- $\{w \in (a + b + c)^* \mid |w|_b = |w|_c\}$, recognised by $\{bc, cb, a, \blacktriangleleft\}$.

Proof of Proposition 74

► **Proposition 74** (Union). *None of the keyboard language classes are stable by union.*

To start with, we consider the languages a^* and b^* , both in MK, and prove that their union is neither in BLEK nor in EAK.

► **Lemma 79.** *The language $L = a^* + b^*$ is not in BLEK.*

Proof. Suppose there exists a keyboard $K = (T, F)$ of BLEK recognising L . Then there exists $\tau_a \in T^*$, $f_a \in F$ such that $\langle \varepsilon | \varepsilon \rangle \cdot \tau_a f_a = \langle u_a | v_a \rangle$ with $u_a v_a = a$.

There also exists $\tau_b \in T^*$, $f_b \in F$ such that $\langle \varepsilon | \varepsilon \rangle \cdot \tau_b f_b = \langle u_b | v_b \rangle$ with $u_b v_b = b^{1+\|K\|_\infty(|\tau_a|+2)}$.

By Lemma 24, applying τ_b to $\langle \varepsilon | \varepsilon \rangle$ yields a configuration with at least $1 + \|K\|_\infty(|\tau_a| + 1)$ b . We apply $\tau_b \tau_a f_a$ to $\langle \varepsilon | \varepsilon \rangle$, by Theorem 56 the resulting configuration contains an a , and as $\tau_a f_a$ can only erase at most $(|\tau_a| + 1)\|K\|_\infty$ letters, it contains a b . This is impossible, as the resulting word should be in L . ◀

► **Lemma 80.** *The language $L = a^* + b^*$ is not in EAK.*

Proof. Suppose there exists $K = (T, F)$ a keyboard of EAK recognising L . As $a^{\|K\|_\infty+1}$ and $b^{\|K\|_\infty+1}$ are both in L , there exist $t_a, t_b \in T$ such that t_a writes an a and t_b a b (and those letters are never erased, as we do not have \leftarrow). Let $f \in F$, $t_a t_b f$ writes a word containing both a and b , thus not in L . ◀

To finish, we define the language $L = L_a \cup L_b$ with $L_a = \{a^n c a^n \mid n \in \mathbb{N}\}$ and $L_b = \{b^n c b^n \mid n \in \mathbb{N}\}$. We can prove that L is not in BEAK, showing that all BAK and BEAK are not stable by intersection.

Proof of Proposition 75

► **Proposition 75** (Intersection emptiness problem). *The following problem is undecidable:*

Input: K_1, K_2 two LK keyboards.

Output: Is $\mathcal{L}(K_1) \cap \mathcal{L}(K_2)$ empty?

Proof. We reduce the Post Correspondence Problem. Let $(u_i, v_i)_{i \in \llbracket 1, n \rrbracket}$ be a PCP instance. For all $i \in \llbracket 1, n \rrbracket$, let u_i^\blacklozenge and v_i^\blacklozenge be u_i and v_i where we added a \blacklozenge at the right of every letter, i.e., if $u_i = a_1 a_2 \cdots a_n$ then $u_i^\blacklozenge = a_1 \blacklozenge a_2 \blacklozenge \cdots a_n \blacklozenge$.

We set for all $i \in \llbracket 1, n \rrbracket$, $t_i = u_i^\blacklozenge \widetilde{v_i^\blacklozenge} \blacktriangleleft^{2|v_i|}$. Let

$$K_{pal} = \{a a \blacktriangleleft \mid a \in A \cup \{\blacklozenge\}\} \cup \{\varepsilon\}$$

K_{pal} recognises the language of even palindromes over $A \cup \{\blacklozenge\}$. Now let

$$K = \{t_i \mid i \in \llbracket 1, n \rrbracket\}.$$

We easily show that $\mathcal{L}(K) \cap \mathcal{L}(K_{pal}) \neq \emptyset$ if and only if $(u_i, v_i)_{i \in \llbracket 1, n \rrbracket} \in \text{PCP}$. ◀

Quantum Speedups for Dynamic Programming on n -Dimensional Lattice Graphs

Adam Glos ✉ 🏠 

Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Warsaw, Poland

Martins Kokainis ✉ 

Centre for Quantum Computer Science, Faculty of Computing, University of Latvia, Riga, Latvia

Ryuhei Mori ✉ 🏠 

School of Computing, Tokyo Institute of Technology, Japan

Jevgēnijs Vihrovs ✉ 

Center for Quantum Computer Science, Faculty of Computing, University of Latvia, Riga, Latvia

Abstract

Motivated by the quantum speedup for dynamic programming on the Boolean hypercube by Ambainis et al. (2019), we investigate which graphs admit a similar quantum advantage. In this paper, we examine a generalization of the Boolean hypercube graph, the n -dimensional lattice graph $Q(D, n)$ with vertices in $\{0, 1, \dots, D\}^n$. We study the complexity of the following problem: given a subgraph G of $Q(D, n)$ via query access to the edges, determine whether there is a path from 0^n to D^n . While the classical query complexity is $\tilde{\Theta}((D+1)^n)$, we show a quantum algorithm with complexity $\tilde{O}(T_D^n)$, where $T_D < D+1$. The first few values of T_D are $T_1 \approx 1.817$, $T_2 \approx 2.660$, $T_3 \approx 3.529$, $T_4 \approx 4.421$, $T_5 \approx 5.332$. We also prove that $T_D \geq \frac{D+1}{e}$ (here, $e \approx 2.718$ is the Euler's number), thus for general D , this algorithm does not provide, for example, a speedup, polynomial in the size of the lattice.

While the presented quantum algorithm is a natural generalization of the known quantum algorithm for $D=1$ by Ambainis et al., the analysis of complexity is rather complicated. For the precise analysis, we use the saddle-point method, which is a common tool in analytic combinatorics, but has not been widely used in this field.

We then show an implementation of this algorithm with time and space complexity $\text{poly}(n)^{\log^n T_D^n}$ in the QRAM model, and apply it to the SET MULTICOVER problem. In this problem, m subsets of $[n]$ are given, and the task is to find the smallest number of these subsets that cover each element of $[n]$ at least D times. While the time complexity of the best known classical algorithm is $O(m(D+1)^n)$, the time complexity of our quantum algorithm is $\text{poly}(m, n)^{\log^n T_D^n}$.

2012 ACM Subject Classification Theory of computation \rightarrow Quantum query complexity; Theory of computation \rightarrow Dynamic programming

Keywords and phrases Quantum query complexity, Dynamic programming, Lattice graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.50

Related Version *Full Version:* <https://arxiv.org/abs/2104.14384>

Funding *Adam Glos:* Supported in part by National Science Center under grant agreement 2019/32/T/ST6/00158 and 2019/33/B/ST6/02011.

Martins Kokainis: Supported by “QuantERA ERA-NET Cofund in Quantum Technologies implemented within the European Union’s Horizon 2020 Programme” (QuantAlgo project).

Ryuhei Mori: Supported in part by JST PRESTO Grant Number JPMJPR1867 and JSPS KAKENHI Grant Numbers JP17K17711, JP18H04090, JP20H04138, and JP20H05966.

Jevgēnijs Vihrovs: Supported in part by the project “Quantum algorithms: from complexity theory to experiment” funded under ERDF programme 1.1.1.5.

Acknowledgements We would like to thank Krišjānis Prūsis for helpful discussions and comments. We also thank anonymous reviewers for helpful comments and suggestions on the presentation.



© Adam Glos, Martins Kokainis, Ryuhei Mori, and Jevgēnijs Vihrovs; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 50; pp. 50:1–50:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Dynamic programming (DP) algorithms have been widely used to solve various NP-hard problems in exponential time. Bellman, Held and Karp showed how DP can be used to solve the TRAVELLING SALESMAN PROBLEM in $\tilde{O}(2^n)^1$ time using DP [5, 22], which still remains the most efficient classical algorithm for this problem. Their technique can be used to solve a plethora of different problems [16, 7].

The DP approach of Bellman, Held and Karp solves the subproblems corresponding to subsets of an n -element set, sequentially in increasing order of the subset size. This typically results in an $\tilde{\Theta}(2^n)$ time algorithm, as there are 2^n distinct subsets. What kind of speedups can we obtain for such algorithms using quantum computers?

It is natural to consider applying Grover's search, which is known to speed up some algorithms for NP-complete problems. For example, we can use it to search through the 2^n possible assignments to the SAT problem instance on n variables in $\tilde{O}(\sqrt{2^n})$ time. However, it is not immediately clear how to apply it to the DP algorithm described above. Recently, Ambainis et al. showed a quantum algorithm that combines classical precalculation with recursive applications of Grover's search that solves such DP problems in $\tilde{O}(1.817^n)$ time, assuming the QRAM model of computation [4].

In their work, the authors applied this result to obtain quantum speedups for the algorithms solving graph vertex ordering problems like PATHWIDTH and SUM CUT [7], and using a more involved analysis, for the GRAPH BANDWIDTH problem [12]. They also used similar ideas to provide speedups for the TRAVELLING SALESMAN, FEEDBACK ARC SET and MINIMUM SET COVER problems by combining the Divide & Conquer and DP techniques. Subsequently, these ideas have been used to construct quantum speedups for the GRAPH COLORING [29], MINIMUM STEINER TREE [26] and finding the optimal variable ordering for the binary decision diagrams (OBDDs) [30]. More surprisingly, [1] used the quantum speedup for the MINIMUM SET COVER to prove non-trivial conditional lower bounds for the k -SUM problem [1] (assuming the Set Cover Conjecture, which states that Minimum Set Cover cannot be solved classically in time $O((2 - \delta)^n)$ for any $\delta > 0$).

The $\tilde{O}(1.817^n)$ quantum speedup of Ambainis et al. for the aforementioned DP algorithm on the subsets of the n -element set examines the underlying transition graph, which can be seen as a directed n -dimensional Boolean hypercube, with edges connecting smaller weight vertices to larger weight vertices. A natural question arises, for what other graphs there exist quantum algorithms that achieve a speedup over the classical DP? In this work, we examine a generalization of the hypercube graph, the n -dimensional lattice graph with vertices in $\{0, 1, \dots, D\}^n$.

While the classical DP for this graph has running time $\tilde{\Theta}((D + 1)^n)$, as it examines all vertices, we prove that there exists a quantum algorithm (in the QRAM model) that solves this problem in time and space $\text{poly}(n)^{\log n} T_D^n$ for $T_D < D + 1$ (Theorems 4, 7). Our algorithm essentially is a generalization of the algorithm of Ambainis et al. We show the following running time for small values of D :

■ **Table 1** The complexity of the quantum algorithm.

D	1	2	3	4	5	6
T_D	1.81692	2.65908	3.52836	4.42064	5.33149	6.25720

¹ $f(n) = \tilde{O}(g(n))$ if $f(n) = O(\log^c(g(n))g(n))$ for some constant c .

A detailed summary of our numerical results is given in Section 5.3. Note that the case $D = 1$ corresponds to the hypercube, where we have the same algorithm and complexity as Ambainis et al. In our proofs, we extensively use the saddle point method from analytic combinatorics to estimate the asymptotic value of the combinatorial expressions arising from the complexity analysis.

It is interesting to compare our analysis and that of Ambainis et al. Their original algorithm is recursive, and solves instances of the problem of smaller size (on the subcubes of smaller dimensions). These instances are symmetrical, so the recursive calls can be analyzed identically, and the calculation of the complexity is relatively simple. However, this is not the case in our generalization for the n -dimensional lattice. To see this, consider, for example, the lattice $\{0, 1, 2\}^n$. In the recursive calls, our algorithm will examine sublattices $\{0, 1\}^{n_1} \times \{0, 1, 2\}^{n_2}$ with fixed maximum vertex weight $w = n_1 + 2n_2$. The first obstacle is that now there are many different cases (n_1, n_2) to analyze. The second obstacle is that recursively we have to solve the problem for a lattice $\{0, 1\}^{n_1} \times \{0, 1, 2\}^{n_2}$: now it becomes difficult to describe and analyze the sublattices examined in the recursion of depth at least 2. To solve these issues, we first make an ansatz that the exponential complexity of the algorithm on the lattice $\times_{d=1}^D \{0, 1, \dots, d\}^{n_d}$ can be expressed as $T_1^{n_1} T_2^{n_2} \dots T_D^{n_D}$, for some positive constants T_d . Then we make use of the saddle point method to find such optimal constants (that minimize T_D), and also prove that the ansatz is correct. Our analysis provides exactly the same $\tilde{O}(1.816\dots^n)$ complexity for the hypercube algorithm as by Ambainis et al.

Next, we also prove a lower bound on the query complexity of the algorithm for general D . Our motivation is to check whether our algorithm, for example, could achieve complexity $\tilde{O}((D+1)^{cn})$ for large D for some $c < 1$. We prove that this is not the case: more specifically, for any D , the algorithm performs at least $\tilde{\Omega}\left(\left(\frac{D+1}{e}\right)^n\right)$ queries (Theorem 5), where $e = 2.71828\dots$ is the Euler's number.

As an example application, we apply our algorithm to the SET MULTICOVER problem (SMC), which is a generalization of the SET COVER problem. In this problem, the input consists of m subsets of the n -element set, and the task is to calculate the smallest number of these subsets that together cover each element at least D times, possibly with overlap and repetition. While the best known classical algorithm has running time $O(m(D+1)^n)$ [27, 24], our quantum algorithm has running time $\text{poly}(m, n)^{\log^n T_D^n}$, improving the exponential complexity (Theorem 8).

The paper is organized as follows. In Section 2, we formally introduce the n -dimensional lattice graph and some of the notation used in the paper. In Section 3, we define the generic query problem that models the examined DP. In Section 4, we describe our quantum algorithm. In Section 5, we establish the query complexity of this algorithm and prove the aforementioned lower bound. In Section 6, we discuss the implementation of this algorithm and establish its time complexity. Finally, in Section 7, we show how to apply our algorithm to SMC, and discuss other related problems.

2 Preliminaries

The n -dimensional lattice graph is defined as follows. The vertex set is given by $\{0, 1, \dots, D\}^n$, and the edge set consists of directed pairs of two vertices u and v such that $v_i = u_i + 1$ for exactly one i , and $u_j = v_j$ for $j \neq i$. We denote this graph by $Q(D, n)$. Alternatively, this graph can be seen as the Cartesian product of n paths on $D + 1$ vertices. The case $D = 1$ is known as the Boolean hypercube and is usually denoted by Q_n .

We define the *weight* of a vertex $x \in V$ as the sum of its coordinates $|x| := \sum_{i=1}^n x_i$. Denote $x \leq y$ iff for all $i \in [n]$, $x_i \leq y_i$ holds. If additionally $x \neq y$, denote such relation by $x < y$.

Throughout the paper we use the standard notation $[n] := \{1, \dots, n\}$. In Section 7.1, we use notation for the superset $2^{[n]} := \{S \mid S \subseteq [n]\}$ and for the characteristic vector $\chi(S) \in \{0, 1\}^n$ of a set $S \in [n]$ defined as $\chi(S)_i = 1$ iff $i \in S$, and 0 otherwise.

We write $f(n) = \text{poly}(n)$ to denote that $f(n) = O(n^c)$ for some constant c . We also write $f(n, m) = \text{poly}(n, m)$ to denote that $f(n, m) = O(n^c m^d)$ for some constants c and d .

For a multivariable polynomial $p(x_1, \dots, x_m)$, we denote by $[x_1^{c_1} \cdots x_m^{c_m}]p(x_1, \dots, x_m)$ its coefficient at the multinomial $x_1^{c_1} \cdots x_m^{c_m}$.

3 Path in the hyperlattice

We formulate our generic problem as follows. The input to the problem is a subgraph G of $Q(D, n)$. The problem is to determine whether there is a path from 0^n to D^n in G . We examine this as a query problem: a single query determines whether an edge (u, v) is present in G or not.

Classically, we can solve this problem using a dynamic programming algorithm that computes the value $\text{dp}(v)$ recursively for all v , which is defined as 1 if there is a path from 0^n to v , and 0 otherwise. It is calculated by the Bellman, Held and Karp style recurrence [5, 22]:

$$\text{dp}(v) = \bigvee_{(u,v) \in E} \{\text{dp}(u) \wedge ((u, v) \in G)\}, \quad \text{dp}(0^n) = 1.$$

The query complexity of this algorithm is $O(n(D+1)^n)$. From this moment we refer to this as the *classical dynamic programming algorithm*.

The query complexity is also lower bounded by $\tilde{\Omega}((D+1)^n)$. Consider the sets of edges E_W connecting the vertices with weights W and $W+1$,

$$E_W := \{(u, v) \mid (u, v) \in Q(D, n), |u| = W, |v| = W+1\}.$$

Since the total number of edges is equal to $(D+1)^{n-1}Dn$, there is such a W that $|E_W| \geq (D+1)^{n-1}Dn/Dn = (D+1)^{n-1}$ (in fact, one can prove that the largest size is achieved for $W = \lfloor nD/2 \rfloor$ [13], but it is not necessary for this argument). Any such E_W is a cut of H_D , hence any path from 0^n to D^n passes through E_W . Examine all G that contain exactly one edge from E_W , and all other edges. Also examine the graph that contains no edges from E_W , and all other edges. In the first case, any such graph contains a desired path, and in the second case there is no such path. To distinguish these cases, one must solve the OR problem on $|E_W|$ variables. Classically, $\Omega(|E_W|)$ queries are needed (see, for example, [8]). Hence, the classical (deterministic and randomized) query complexity of this problem is $\tilde{\Theta}((D+1)^n)$. This also implies $\tilde{\Omega}(\sqrt{(D+1)^n})$ quantum lower bound for this problem [6].

4 The quantum algorithm

Our algorithm closely follows the ideas of [4]. We will use the well-known generalization of Grover's search:

► **Theorem 1** (Variable time quantum search (VTS), Theorem 3 in [3]). *Let $\mathcal{A}_1, \dots, \mathcal{A}_N$ be quantum algorithms that compute a function $f : [N] \rightarrow \{0, 1\}$ and have query complexities t_1, \dots, t_N , respectively, which are known beforehand. Suppose that for each \mathcal{A}_i , if $f(i) = 0$, then $\mathcal{A}_i = 0$ with certainty, and if $f(i) = 1$, then $\mathcal{A}_i = 1$ with constant success probability. Then there exists a quantum algorithm with constant success probability that checks whether $f(i) = 1$ for at least one i and has query complexity $O\left(\sqrt{t_1^2 + \dots + t_N^2}\right)$. Moreover, if $f(i) = 0$ for all $i \in [N]$, then the algorithm outputs 0 with certainty.*

Even though Ambainis formulates the main theorem for zero-error inputs, the statement above follows from the construction of the algorithm.

Now we describe our algorithm. We solve a more general problem: suppose $s, t \in \{0, 1, \dots, D\}^n$ are such that $s < t$ and we are given a subgraph of the n -dimensional lattice

$$\times_{i=1}^n \{s_i, \dots, t_i\},$$

and the task is to determine whether there is path from s to t . We need this generalized problem because our algorithm is recursive and is called for sublattices.

Define $d_i := t_i - s_i$. Let n_d be the number of indices $i \in [n]$ such that $d_i = d$. Note that the minimum and maximum weights of the vertices of this lattice are $|s|$ and $|t|$, respectively.

We call a set of vertices with fixed total weight a *layer*. The algorithm will operate with K layers (numbered 1 to K), with the k -th having weight $|s| + W_k$, where $W_k := \left\lfloor \sum_{d=1}^D \alpha_{k,d} n_d \right\rfloor$. Denote the set of vertices in this layer by

$$\mathcal{L}_k := \{v \mid |v| = |s| + W_k\}.$$

Here, $\alpha_{k,d} \in (0, 1/2)$ are constant parameters that have to be determined before we run the algorithm. The choice of $\alpha_{k,d}$ does not depend on the input to the algorithm, similarly as it was in [4]. For each $k \in [K]$ and $d \in [D]$, we require that $\alpha_{k,d} < \alpha_{k+1,d}$. In addition to the K layers defined in this way, we also consider the $(K+1)$ -th layer \mathcal{L}_{K+1} , which is the set of vertices with weight $|s| + W_{K+1}$, where $W_{K+1} := \left\lfloor \frac{|t| - |s|}{2} \right\rfloor$. We can see that the weights W_1, \dots, W_{K+1} defined in this way are non-decreasing.

The informal description of the algorithm (PATH) is as follows. First, we use the classical dynamic programming to calculate which vertices v with weight $|v| \leq |s| + W_1$ are reachable from s . Then, we store all of these answers in memory. Symmetrically, we also calculate from which vertices v with weight $|v| \geq |t| - W_1$ we can reach t , and also store this in memory. We refer to these steps as the classical precalculation part.

Next, we use VTS to search for a vertex $v^{(K+1)}$ in the layer \mathcal{L}_{K+1} such that there is path from s to $v^{(K+1)}$ and from v to t . The LAYERPATH function is then used to detect whether there is a path from s to $v^{(K+1)}$. First, we use VTS to search for a vertex $v^{(K)} \in \mathcal{L}_K$ such that: (1) there exists a path from $v^{(K)}$ to $v^{(K+1)}$; (2) there exists a path from s to $v^{(K)}$. The first condition we can check using PATH recursively for the lattice bounded by the vertices $v^{(K)}$ and $v^{(K+1)}$. The second condition is checked recursively using LAYERPATH in a similar fashion. Finally, for the vertex $v^{(1)} \in \mathcal{L}_1$, the LAYERPATH will need to check whether there is a path from s to $v^{(1)}$: this can be then simply read out from the memory, using the results of the precalculation part. We then similarly find whether t is reachable from $v^{(K)}$.

5 Query complexity

For simplicity, let us examine the lattice

$$\times_{i=1}^n \{0, \dots, t_i - s_i\},$$

as the analysis is identical.

Let the number of positions with maximum coordinate value d be n_d . We make an ansatz that the exponential complexity can be expressed as

$$T(n_1, \dots, n_D) := T_1^{n_1} T_2^{n_2} \cdot \dots \cdot T_D^{n_D}$$

■ **Algorithm 1** The quantum algorithm for detecting a path in the hyperlattice.

PATH(s, t):

1. Calculate n_1, \dots, n_D , and W_1, \dots, W_{K+1} . If $W_k = W_{k+1}$ for some k , determine whether there exists a path from s to t using classical dynamic programming and return.
2. Otherwise, first perform the precalculation step. Let $\text{dp}(v)$ be 1 iff there is a path from s to v . Calculate $\text{dp}(v)$ for all vertices v such that $|v| \leq |s| + W_1$ using classical dynamic programming. Store the values of $\text{dp}(v)$ for all vertices with $|v| = |s| + W_1$. Let $\text{dp}'(v)$ be 1 iff there is a path from v to t . Symmetrically, we also calculate $\text{dp}'(v)$ for all vertices with $|v| = |t| - W_1$.
3. Define the function $\text{LAYERPATH}(k, v)$ to be 1 iff there is a path from s to v such that $v \in \mathcal{L}_k$. Implement this function recursively as follows.
 - $\text{LAYERPATH}(1, v)$ is read out from the stored values.
 - For $k > 1$, run VTS over the vertices $u \in \mathcal{L}_{k-1}$ such that $u < v$. The required value is equal to

$$\text{LAYERPATH}(k, v) = \bigvee_u \{\text{LAYERPATH}(k-1, u) \wedge \text{PATH}(u, v)\}.$$

4. Similarly define and implement the function $\text{LAYERPATH}'(k, v)$, which denotes the existence of a path from v to t such that $v \in \mathcal{L}'_k$ (where \mathcal{L}'_k is the layer with weight $|t| - W_k$). To find the final answer, run VTS over the vertices in the middle layer $v \in \mathcal{L}_{K+1}$ and calculate

$$\bigvee_v \{\text{LAYERPATH}(K+1, v) \wedge \text{LAYERPATH}'(K+1, v)\}.$$

for some values $T_1, T_2, \dots, T_D > 1$ (we also can include n_0 and T_0 , however, $T_0 = 1$ always and doesn't affect the complexity). We prove it by constructing generating polynomials for the precalculation and quantum search steps, and then approximating the required coefficients asymptotically. We use the saddle point method that is frequently used for such estimation, specifically the theorems developed in [9].

5.1 Generating polynomials

First we estimate the number of edges of the hyperlattice queried in the precalculation step. The algorithm queries edges incoming to the vertices of weight at most W_1 , and each vertex can have at most n incoming edges. The size of any layer with weight less than W_1 is at most the size of the layer with weight exactly W_1 , as the size of the layers is non-decreasing until weight W_{K+1} [13]. Therefore, the number of queries during the precalculation is at most $n \cdot W_1 \cdot |\mathcal{L}_1| \leq n^2 D |\mathcal{L}_1|$, as $W_1 \leq nD$. Since we are interested in the exponential complexity, we can omit n and D , thus the exponential query complexity of the precalculation is given by $|\mathcal{L}_1|$.

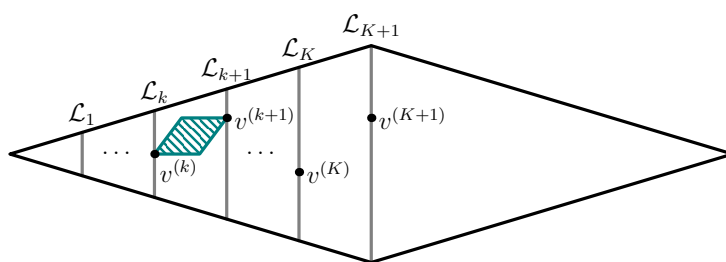
Now let $P_d(x) := \sum_{i=0}^d x^i$. The number of vertices of weight W_1 can be written as the coefficient at x^{W_1} of the generating polynomial

$$P(x) := \prod_{d=0}^D P_d(x)^{n_d}.$$

Indeed, each $P_d(x)$ in the product corresponds to a single position $i \in [n]$ with maximum value d and the power of x in that factor represents the coordinate of the vertex in this position. Therefore, the total power that x is raised to is equal to the total weight of the vertex, and coefficient at x^{W_1} is equal to the number of vertices with weight W_1 . Since the total query complexity of the algorithm is lower bounded by this coefficient, we have

$$T(n_1, \dots, n_D) \geq [x^{W_1}]P(x). \tag{1}$$

Similarly, we construct polynomials for the LAYERPATH calls. Consider the total complexity of calling LAYERPATH recursively until some level $1 \leq k \leq K$ and then calling PATH for a sublattice between levels \mathcal{L}_k and \mathcal{L}_{k+1} . Define the variables for the vertices chosen by the algorithm at level i (where $k \leq i \leq K + 1$) by $v^{(i)}$. The PATH call is performed on a sublattice between vertices $v^{(k)}$ and $v^{(k+1)}$, see Fig. 1.



■ **Figure 1** The choice of the vertices $v^{(i)}$ and the application of PATH on the sublattice.

Define

$$S_{k,d}(x_{k,k}, \dots, x_{k,K+1}) := \sum_{i=0}^d T_i^2 \cdot \sum_{\substack{p_k, \dots, p_{K+1} \in [0,d] \\ p_{k+1} \leq \dots \leq p_{K+1} \\ p_{k+1} - p_k = i}} \prod_{j=k}^{K+1} x_{k,j}^{p_j}.$$

Again, this corresponds to a single coordinate. The variable $x_{k,j}$ corresponds to the vertex $v^{(j)}$ and the power p_j corresponds to the value of $v^{(j)}$ in that coordinate.

Examine the following multivariate polynomial:

$$S_k(x_{k,k}, \dots, x_{k,K+1}) := \prod_{d=0}^D S_{k,d}^{m_d}(x_{k,k}, \dots, x_{k,K+1}).$$

We claim that the coefficient

$$[x_{k,k}^{W_k} \cdots x_{k,K+1}^{W_{K+1}}] S_k(x_{k,k}, \dots, x_{k,K+1})$$

is the required total complexity squared.

First of all, note that the value of this coefficient is the sum of t^2 , where t is the variable for the running time of PATH between $v^{(k)}$ and $v^{(k+1)}$, for all choices of vertices $v^{(k)}, v^{(k+1)}, \dots, v^{(K+1)}$. Indeed, the powers p_j encode the values of coordinates of $v^{(j)}$, and a factor of T_i^2 is present for each multinomial that has $p_{k+1} - p_k = i$ (that is, $v_i^{(k+1)} - v_i^{(k)} = i$ for the corresponding position i).

Then, we need to show that the sum of t^2 equals the examined running time squared. Note that the choice of each vertex $v^{(j)}$ is performed using VTS. In general, if we perform VTS on the algorithms with running times s_1, \dots, s_N , then the total squared running time

is equal to $s_1^2 + \dots + s_N^2$ by Theorem 1. By repeating this argument in our case inductively at the choice of each vertex $v^{(j)}$, we obtain that the final squared running time indeed is the sum of all t^2 .

Therefore, the square of the total running time of the algorithm is lower bounded by

$$T(n_1, \dots, n_D)^2 \geq \left[x_{k,k}^{W_k} \cdots x_{k,K+1}^{W_{K+1}} \right] S_k(x_{k,k}, \dots, x_{k,K+1}). \quad (2)$$

Together the inequalities (1) and (2) allow us to estimate T . The total time complexity of the quantum algorithm is twice the sum of the coefficients given in Eq. (1) and (2) for all $k \in [K]$ (twice because of the calls to LAYERPATH and its symmetric counterpart LAYERPATH'). This is upper bounded by $2K$ times the maximum of these coefficients. Since $2K$ is a constant, and there are $O(\log n)$ levels of recursion (see Appendix A), in total this contributes only $(2K)^{O(\log n)} = \text{poly}(n)$ factor to the total complexity of the quantum algorithm.

5.2 Saddle point approximation

In this section, we show how to describe the tight asymptotic complexity of $T(n_1, \dots, n_D)$ using the saddle point method (a detailed review can be found in [15], Chapter VIII). Our main technical tool will be the following theorem.

► **Theorem 2.** *Let $p_1(x_1, \dots, x_m), \dots, p_D(x_1, \dots, x_m)$ be polynomials with non-negative coefficients. Let n be a positive integer and b_1, \dots, b_D be non-negative rational numbers such that $b_1 + \dots + b_D = 1$ and $b_d n$ is an integer for all $d \in [D]$. Let $a_{i,d}$ be rational numbers (for $i \in [m], d \in [D]$) and $\alpha_i := a_{i,1} b_1 + \dots + a_{i,D} b_D$. Suppose that $\alpha_i n$ are integer for all $i \in [m]$. Then*

$$(1) \quad [x_1^{\alpha_1 n} \cdots x_m^{\alpha_m n}] \prod_{d=1}^D p_d(x_1, \dots, x_m)^{b_d n} \leq \left(\inf_{x_1, \dots, x_m > 0} \prod_{d=1}^D \left(\frac{p_d(x_1, \dots, x_m)}{x_1^{a_{1,d}} \cdots x_m^{a_{m,d}}} \right)^{b_d} \right)^n$$

$$(2) \quad [x_1^{\alpha_1 n} \cdots x_m^{\alpha_m n}] \prod_{d=1}^D p_d(x_1, \dots, x_m)^{b_d n} = \Omega \left(\left(\inf_{x_1, \dots, x_m > 0} \prod_{d=1}^D \left(\frac{p_d(x_1, \dots, x_m)}{x_1^{a_{1,d}} \cdots x_m^{a_{m,d}}} \right)^{b_d} \right)^n \right),$$

where Ω depends on the variable n .

Proof. To prove this, we use the following saddle point approximation.²

► **Theorem 3** (Saddle point method, Theorem 2 in [9]). *Let $p(x_1, \dots, x_m)$ be a polynomial with non-negative coefficients. Let $\alpha_1, \dots, \alpha_m$ be some rational numbers and let n_i be the series of all integers j such that $\alpha_k j$ are integers and $[x_1^{\alpha_1 j} \cdots x_m^{\alpha_m j}] p(x_1, \dots, x_m)^j \neq 0$. Then*

$$\lim_{i \rightarrow \infty} \frac{1}{n_i} \log([x_1^{\alpha_1 n_i} \cdots x_m^{\alpha_m n_i}] p(x_1, \dots, x_m)^{n_i}) = \inf_{x_1, \dots, x_m > 0} \log \left(\frac{p(x_1, \dots, x_m)}{x_1^{\alpha_1} \cdots x_m^{\alpha_m}} \right).$$

Let $p(x_1, \dots, x_m) := \prod_{d=1}^D p_d(x_1, \dots, x_m)^{b_d}$, then

$$\frac{p(x_1, \dots, x_m)}{x_1^{\alpha_1} \cdots x_m^{\alpha_m}} = \frac{\prod_{d=1}^D p_d(x_1, \dots, x_m)^{b_d}}{x_1^{\alpha_1} \cdots x_m^{\alpha_m}} = \prod_{d=1}^D \frac{p_d(x_1, \dots, x_m)^{b_d}}{x_1^{a_{1,d} b_d} \cdots x_m^{a_{m,d} b_d}} = \prod_{d=1}^D \left(\frac{p_d(x_1, \dots, x_m)}{x_1^{a_{1,d}} \cdots x_m^{a_{m,d}}} \right)^{b_d}.$$

² Setting $\gamma = 1$ in the statement of the original theorem.

For the first part, as $p(x_1, \dots, x_m)^n$ has non-negative coefficients, the coefficient at the multinomial $x_1^{\alpha_1 n} \dots x_m^{\alpha_m n}$ is upper bounded by

$$\inf_{x_1, \dots, x_m > 0} \frac{p(x_1, \dots, x_m)^n}{x_1^{\alpha_1 n} \dots x_m^{\alpha_m n}} = \left(\inf_{x_1, \dots, x_m > 0} \frac{p(x_1, \dots, x_m)}{x_1^{\alpha_1} \dots x_m^{\alpha_m}} \right)^n$$

The second part follows directly by Theorem 3. ◀

5.2.1 Optimization program

To determine the complexity of the algorithm, we construct the following optimization problem. Recall that the Algorithm 1 is given by the number of layers K and the constants $\alpha_{k,d}$ that determine the weight of the layers, so assume they are fixed known numbers. Assume that $\alpha_{k,d}$ are all rational numbers between 0 and $1/2$ for $k \in [K]$; indeed, we can approximate any real number with arbitrary precision by a rational number. Also let $T_0 = 1$ and $\alpha_{K+1,d} = 1/2$ for all $d \in [D]$ for convenience.

Examine the following program $\text{OPT}(D, K, \{\alpha_{k,d}\})$:

$$\begin{aligned} \text{minimize } T_D \quad \text{s.t.} \quad & T_d \geq \frac{P_d(x)}{x^{\alpha_{1,d}}} && \forall d \in [D] \\ & T_d^2 \geq \frac{S_{k,d}(x_{k,k}, \dots, x_{k,K+1})}{x_{k,k}^{\alpha_{k,d}} \dots x_{k,K+1}^{\alpha_{K+1,d}}} && \forall d \in [D], \forall k \in [K] \\ & T_d \geq 1 && \forall d \in [D] \\ & x > 0 \\ & x_{k,j} > 0 && \forall k \in [K], \forall j \in \{k, \dots, K+1\} \end{aligned}$$

Let $n := n_1 + \dots + n_D$ and $\alpha_k := \frac{\sum_{d=1}^D \alpha_{k,d} n_d}{n}$. Suppose that T_1, \dots, T_D is a feasible point of the program. Then by Theorem 2 (1) (setting $b_i := n_i/n$ and $a_{i,d} := \alpha_{i,d} n_d$) we have

$$[x^{\alpha_1 n}]P(x) \leq \inf_{x > 0} \prod_{d=1}^D \left(\frac{P_d(x)}{x^{\alpha_{1,d} n_d}} \right)^{n_d} \leq T_1^{n_1} \dots T_D^{n_D}.$$

Similarly,

$$\begin{aligned} [x_{k,k}^{\alpha_{k,n}} \dots x_{k,K+1}^{\alpha_{K+1,n}}] S_k(x_{k,k}, \dots, x_{k,K+1}) &\leq \inf_{x_{k,k}, \dots, x_{k,K+1} > 0} \prod_{d=1}^D \left(\frac{S_{k,d}(x_{k,k}, \dots, x_{k,K+1})}{x_{k,k}^{\alpha_{k,d} n_d} \dots x_{k,K+1}^{\alpha_{K+1,d} n_d}} \right)^{n_d} \\ &\leq (T_1^{n_1} \dots T_D^{n_D})^2. \end{aligned}$$

Therefore, the program provides an upper bound on the complexity. There are two subtleties that we need to address for correctness: firstly, what happens when $\alpha_k n$ is not an integer; secondly, the case when $W_k = W_{k+1}$ for some k . We show that both do not raise an issue in Appendix B.

5.2.2 Optimality of the program

In the start of the analysis, we made an assumption that the exponential complexity $T(n_1, \dots, n_D)$ can be expressed as $T_1^{n_1} \dots T_D^{n_D}$. In Appendix C, using the lower bound of Theorem 3 (2), we show that $\text{OPT}(D, K, \{\alpha_{k,d}\})$ (which gives an upper bound on the complexity) can indeed achieve such value and also gives the best possible solution.

5.2.3 Total complexity

Finally, in Appendix D we argue that there exists a choice for the parameters $\{\alpha_{k,d}\}$ such that $\text{OPT}(D, K, \{\alpha_{k,d}\}) < D + 1$. Therefore, putting all together, we have the main result:

► **Theorem 4.** *There exists a bounded-error quantum algorithm that solves the path in the n -dimensional lattice problem using $\tilde{O}(T_D^n)$ queries, where $T_D < D + 1$. The optimal value of T_D can be found by optimizing $\text{OPT}(D, K, \{\alpha_{k,d}\})$ over K and $\{\alpha_{k,d}\}$.*

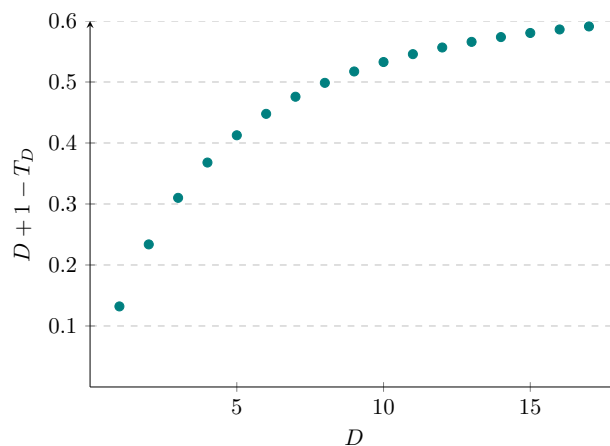
5.3 Complexity for small D

To find the estimate on the complexity for small values of D and K , we have optimized the value of $\text{OPT}(D, K, \{\alpha_{k,d}\})$ using Mathematica (minimizing over the values of $\alpha_{k,d}$). Table 2 compiles the results obtained by the optimization. In case of $D = 1$, we recovered the complexity of the quantum algorithm from [4] for the path in the hypercube problem, which is a special case of our algorithm.

■ **Table 2** The complexity of the quantum algorithm for small values of D and K .

	$D = 1$	$D = 2$	$D = 3$	$D = 4$	$D = 5$	$D = 6$
$K = 1$	1.86793	2.76625	3.68995	4.63206	5.58735	6.55223
$K = 2$	1.82562	2.67843	3.55933	4.46334	5.38554	6.32193
$K = 3$	1.81819	2.66198	3.53322	4.42759	5.34059	6.26840
$K = 4$	1.81707	2.65939	3.52893	4.42148	5.33263	6.25862
$K = 5$	1.81692	2.65908	3.52836	4.42064	5.33149	6.25720

For $K = 1$, we were able to estimate the complexity for up to $D = 18$. Figure 2 shows the values of the difference between $D + 1$ and T_D for this range.



■ **Figure 2** The advantage of the quantum algorithm over the classical for $K = 1$.

Our Mathematica code used for determining the values of T_D can be accessed at <https://doi.org/10.5281/zenodo.4603689>. In Appendix E, we list the parameters for the case $K = 1$.

5.4 Lower bound for general D

Even though Theorem 4 establishes the quantum advantage of the algorithm, it is interesting how large the speedup can get for large D . In this section, we prove that the speedup cannot be substantial, more specifically:

► **Theorem 5.** *For any fixed integers $D \geq 1$ and $K \geq 1$, Algorithm 1 performs $\tilde{\Omega}\left(\left(\frac{D+1}{e}\right)^n\right)$ queries on the lattice $Q(D, n)$.*

Proof. The structure of the proof is as follows. First, we prove that if $\alpha_{1,D} > \frac{1}{4}$, then the number of queries used in the algorithm during the precalculation step 2 is at least $\tilde{\Omega}((0.664554(D+1))^n)$ queries. Then, we prove that if $\alpha_{1,D} \leq \frac{1}{4}$, then the quantum search part in steps 3 and 4 performs at least $\tilde{\Omega}\left(\left(\frac{D+1}{e}\right)^n\right)$ queries. Therefore, depending on whether $\alpha_{1,D} > \frac{1}{4}$, one of the precalculation or the quantum search performs $\tilde{\Omega}((c(D+1))^n)$ queries for constant c , and the claim follows, since $\frac{1}{e} < 0.664554$. Due to space limitations, the proof can be accessed online in the full version of the paper [18] (Appendix B). ◀

6 Time complexity

In this section we examine a possible high-level implementation of the described algorithm and argue that there exists a quantum algorithm with the same exponential time complexity as the query complexity.

Firstly, we assume the commonly used QRAM model of computation that allows to access N memory cells in superposition in time $O(\log N)$ [17]. This is needed when the algorithm accesses the precalculated values of dp . Since in our case N is always at most $(D+1)^n$, this introduces only a $O(\log((D+1)^n)) = O(n)$ additional factor to the time complexity.

The main problem that arises is the efficient implementation of VTS. During the VTS execution, multiple quantum algorithms should be performed in superposition. More formally, to apply VTS to algorithms $\mathcal{A}_1, \dots, \mathcal{A}_N$, we should specify the *algorithm oracle* that, given the index of the algorithm i and the time step t , applies the t -th step of \mathcal{A}_i (see Section 2.2 of [11] for formal definition of such an oracle and related discussion). If the algorithms \mathcal{A}_i are unstructured, the implementation of such an oracle may take even $O(N)$ time (if, for example, all of the algorithms perform a different gate on different qubits at the t -th step).

We circumvent this issue by showing that it is possible to use only Grover's search to implement the algorithm, retaining the same exponential complexity (however, the sub-exponential factor in the complexity will increase). Nonetheless, the use of VTS in the query algorithm not only achieves a smaller query complexity, but also allowed to prove the estimate on the exponential complexity, which would not be so amiable for the algorithm that uses Grover's search.

6.1 Implementation

The main idea of the implementation is to fix a “class” of vertices for each of the $2K+1$ layers examined by the algorithm, and do this for all $r = O(\log n)$ levels of recursion. We will essentially define these classes by the number of coordinates of a vertex in such layer that are equal to 0, 1, \dots , D . Then, we can first fix a class for each layer for all levels of recursion classically. We will show that there are at most n^{D^2} different classes we have to consider at each layer. Since there are $2K+1$ layers at one level of recursion, and $O(\log n)$ levels of recursion, this classical precalculation will take time $n^{O(D^2 K \log n)}$. For each such choice of classes, we will run a quantum algorithm that checks for the path in the hyperlattice constrained on these classes of the vertices the path can go through. The advantage of

the quantum algorithm will come from checking the permutations of the coordinates using Grover’s search. The time complexity of the quantum part will be $n^{O(K \log n)} T_D^n$ (T_D^n as in the query algorithm, and $n^{O(K \log n)}$ from the logarithmic factors in Grover’s search), therefore the total time complexity will be $n^{O(D^2 K \log n)} \cdot n^{O(K \log n)} T_D^n = n^{O(D^2 K \log n)} T_D^n$, thus the exponential complexity stays the same.

6.2 Layer classes

In all of the applications of VTS in the algorithm, we use it in the following scenario: given a vertex x , examine all vertices y with fixed weight $|y| = W$ such that $y < x$ (note that VTS over the middle layer \mathcal{L}_{K+1} can be viewed in this way by taking x to be the final vertex in the lattice, and VTS over the vertices in the layers symmetrical to \mathcal{L}_{K+1} can be analyzed similarly).

We define a *class* of y ’s (in respect to x) in the following way. Let $n_{a,b}$ be the number of $i \in [n]$ such that $y_i = a$ and $x_i = b$, where $a \leq b$. All y in the same class have the same values of $n_{a,b}$ for all a, b . Also define a *representative* of a class as a single particular y from that class; we will define it as the lexicographically smallest such y .

As mentioned in the informal description above, we can fix the classes for all layers examined by the quantum algorithm and generate the corresponding representatives classically. Note that in our quantum algorithm, recursive calls work with the sublattice constrained on the vertices $s \leq y \leq t$ for some $s < t$, so for each position of y_i we should have also $y_i \geq s_i$; however, we can reduce it to lattice $0^n \leq y' \leq x$, where $x_i := t_i - s_i$ for all i . To get the real value of y , we generate a representative y' , and set $y_i := y'_i + s_i$.

Consider an example for $D = 2$. The following figure illustrates the representative y (note that the order of positions of x here is lexicographical for simplicity, but it may be arbitrary).

$$\begin{array}{l}
 x = 00 \dots 011 \dots \dots \dots 122 \dots \dots \dots 2 \\
 y = \underbrace{00 \dots 0}_{n_{0,0}} \underbrace{00 \dots 0}_{n_{0,1}} \underbrace{11 \dots 1}_{n_{1,1}} \underbrace{00 \dots 0}_{n_{0,2}} \underbrace{11 \dots 1}_{n_{1,2}} \underbrace{22 \dots 2}_{n_{2,2}}
 \end{array}$$

■ **Figure 3** The (lexicographically smallest) representative for y for $D = 2$.

Note that $n_{a,b}$ can be at most n . Therefore, there are at most n^{D^2} choices for classes at each layer. Thus the total number of different sets of choices for all layers is $n^{O(D^2 K \log n)}$. For each such set of choices, we then run a quantum algorithm that checks for a path in the sublattice constrained on these classes.

6.3 Quantum algorithm

The algorithm basically implements Algorithm 1, with VTS replaced by Grover’s search. Thus we only describe how we run the Grover’s search. We will also use the analysis of Grover’s search with multiple marked elements.

► **Theorem 6** (Grover’s search). *Let $f : S \rightarrow \{0, 1\}$, where $|S| = N$. Suppose we can generate a uniform superposition $\frac{1}{\sqrt{N}} \sum_{x \in S} |x\rangle$ in $O(\text{poly}(\log N))$ time, and there is a bounded-error quantum algorithm \mathcal{A} that computes $f(x)$ with time complexity T . Suppose also that there is a promise that either there are at least k solutions to $f(x) = 1$, or there are none. Then there exists a bounded-error quantum algorithm that runs in time $O(T \log N \sqrt{N/k})$, and detects whether there exists x such that $f(x) = 1$.*

Proof. First, it is well-known that in the case of k marked elements, Grover's algorithm [21] needs $O(\sqrt{N/k})$ iterations. Second, the gate complexity of one iteration of Grover's search is known to be $O(\log N)$. Finally, even though \mathcal{A} has constant probability of error, there is a result that implements Grover's search with a bounded-error oracle without introducing another logarithmic factor [23]. ◀

Now, for a class \mathcal{C} of y 's (for a fixed x) we need to generate a superposition $\frac{1}{\sqrt{|\mathcal{C}|}} \sum_{y \in \mathcal{C}} |y\rangle$ efficiently to apply Grover's algorithm. We will generate a slightly different superposition for the same purposes. Let I_1, \dots, I_D be sets $I_d := \{i \in [n] \mid x_i = d\}$ and let $n_d := |I_d|$. Let $y_{\mathcal{C}}$ be the representative of \mathcal{C} . We will generate the superposition

$$\bigotimes_{d=0}^D \frac{1}{\sqrt{n_d!}} \sum_{\pi \in S_{n_d}} |\pi(y_{\mathcal{C}_{I_d}})\rangle |\pi\rangle, \quad (3)$$

where $y_{\mathcal{C}_{I_d}}$ are the positions of $y_{\mathcal{C}}$ in I_d .

We need a couple of procedures to generate such state. First, there exists a procedure to generate the uniform superposition of permutations $\frac{1}{\sqrt{n!}} \sum_{\pi \in S_n} |\pi_1, \dots, \pi_n\rangle$ that requires $O(n^2 \log n)$ elementary gates [2, 10]. Then, we can build a circuit with $O(\text{poly}(n))$ gates that takes as an input $\pi \in S_n$, $s \in \{0, 1, \dots, D\}^n$ and returns $\pi(s)$. Such a circuit essentially could work as follows: let $t := 0^n$; then for each pair $i, j \in [n]$, check whether $\pi(i) = j$; if yes, let $t_j \leftarrow t_j + s_{\pi(i)}$; in the end return t . Using these two subroutines, we can generate the required superposition using $O(\text{poly}(n))$ gates (we assume D is a constant).

However, we do not necessarily know the sets I_d , because the positions of x have been permuted by previous applications of permutations. To mitigate this, note that we can access this permutation in its own register from the previous computation. That is, suppose that x belongs to a class \mathcal{C}' and $x = \sigma(x_{\mathcal{C}'})$, where $x_{\mathcal{C}'}$ is the representative of \mathcal{C}' generated by the classical algorithm from the previous subsection. Then we have the state $|\sigma(x_{\mathcal{C}'})\rangle |\sigma\rangle$.

We can then apply σ to both $\pi(y_{\mathcal{C}})$ and π . That is, we implement the transformation

$$|\pi(y_{\mathcal{C}})\rangle |\pi\rangle \rightarrow |\sigma(\pi(y_{\mathcal{C}}))\rangle |\sigma\pi\rangle.$$

Such transformation can also be implemented in $O(\text{poly}(n))$ gates. Note that now we store the permutation $\sigma\pi$ in a separate register, which we use in a similar way recursively.

Finally, examine the number of positive solutions among $\pi(y_{\mathcal{C}})$. That is, for how many π there exists a path from $\pi(y)$ to x ? Suppose that there is a path from y to x for some $y \in \mathcal{C}$. Examine the indices I_d ; for $n_{a,d}$ of these indices i we have $y_i = a$. There are exactly $n_{a,d}!$ permutations that permute these indices and don't change y . Hence, there are $\prod_{a=0}^D n_{a,d}!$ distinct permutations $\pi \in S_{n_d}$ such that $\pi(y) = y$.

Therefore, there are $k := \prod_{d=0}^D \prod_{a=0}^d n_{a,d}!$ distinct permutations π among the considered such that $\pi(y) = y$. The total number of considered permutations is $N := \prod_{d=0}^D n_d!$. Among these permutations, either there are no positive solutions, or at least k of the solutions are positive. Grover's search then works in time $O(T \log N \sqrt{N/k})$. In this case, N/k is exactly the size of the class \mathcal{C} , because $\frac{n_d!}{n_{0,d}! \dots n_{d,d}!}$ is the number of unique permutations of $y_{\mathcal{C}_{P_d}}$, the multinomial coefficient $\binom{n_d}{n_{0,d}, \dots, n_{d,d}}$. Hence the state Eq. (3) effectively replaces the need for the state $\frac{1}{\sqrt{|\mathcal{C}|}} \sum_{y \in \mathcal{C}} |y\rangle$.

6.4 Total complexity

Finally, we discuss the total time complexity of this algorithm. The exponential time complexity of the described quantum algorithm is at most the exponential query complexity because Grover's search examines a single class \mathcal{C} , while VTS in the query algorithm examines

all possible classes. Since Grover's search has a logarithmic factor overhead, the total time complexity of the quantum part of the algorithm is what is described in Section 5 multiplied by $n^{O(K \log n)}$, resulting in $n^{O(K \log n)} T_1^{n_1} \dots T_D^{n_D}$.

Since there are $n^{O(D^2 K \log n)}$ sets of choices for the classes of the layers, the final total time complexity of the algorithm is $n^{O(D^2 K \log n)} T_1^{n_1} \dots T_D^{n_D}$.

For the space complexity, note that the precalculation step requires asymptotically the same exponential amount of space as time, thus $T_1^{n_1} \dots T_D^{n_D}$ is also the exponential space complexity of the algorithm.

Therefore, we have the following result.

► **Theorem 7.** *Assuming QRAM model of computation, there exists a quantum algorithm that solves the path in the n -dimensional lattice problem with time and space complexity $\text{poly}(n)^{D^2 \log n} \cdot T_D^n$.*

7 Applications

7.1 Set multicover

As an example application of our algorithm, we apply it to the SET MULTICOVER problem (SMC). This is a generalization of the MINIMUM SET COVER problem. The SMC problem is formulated as follows:

Input: A set of subsets $\mathcal{S} \subseteq 2^{[n]}$, and a positive integer D .

Output: The size k of the smallest tuple $(S_1, \dots, S_k) \in \mathcal{S}^k$, such that for all $i \in [n]$, we have $|\{j \mid i \in S_j\}| \geq D$, that is, each element is covered at least D times (note that each set $S \in \mathcal{S}$ can be used more than once).

Denote this problem by SMC_D , and $m := |\mathcal{S}|$. This problem has been studied classically, and there exists an exact deterministic algorithm based on the inclusion-exclusion principle that solves this problem in time $\tilde{O}(m(D+1)^n)$ and polynomial space [27, 24]. While there are various approximation algorithms for this problem, we are not aware of a more efficient classical exact algorithm.

There is a different simple classical dynamic programming algorithm for this problem with the same time complexity (although it uses exponential space), which we can speed up using our quantum algorithm. For a vector $x \in \{0, 1, \dots, D\}^n$, define $\text{dp}(x)$ to be the size k of the smallest tuple $(\mathcal{C}_1, \dots, \mathcal{C}_k) \in \mathcal{S}^k$ such that for each i , we have $|\{j \in [k] \mid i \in \mathcal{C}_j\}| \geq x_i$. It can be calculated using the recurrence

$$\text{dp}(0^n) = 0, \quad \text{dp}(x) = 1 + \min_{S \in \mathcal{S}} \{\text{dp}(x')\},$$

where x' is given by $x'_i = \max\{0, x_i - \chi(S)_i\}$ for all i . Consequently, the answer to the problem is equal to $\text{dp}(D^n)$. The number of distinct x is $(D+1)^n$, and $\text{dp}(x)$ for a single x can be calculated in time $O(nm)$, if $\text{dp}(y)$ has been calculated for all $y < x$. Thus the time complexity is $O(nm(D+1)^n)$ and space complexity is $O((D+1)^n)$.

Note that even though the state space of the dynamic programming here is $\{0, 1, \dots, D\}^n$, the underlying transition graph is not the same as the hyperlattice examined in the quantum algorithm. A set $S \in \mathcal{S}$ can connect vertices that are $|S|$ distance apart from each other, unlike distance 1 in the hyperlattice. We can essentially reduce this to the hyperlattice-like transition graph by breaking such transition into $|S|$ distinct transitions.

More formally, examine pairs (x, S) , where $x \in \{0, 1, \dots, D\}^n$, $S \in \mathcal{S}$. Let $e(x, S) := \min\{i \in S \mid x_i > 0\}$; if there is no such i , let $e(x, S)$ be 0. Define a new function

$$\text{dp}(x, S) = \begin{cases} 0, & \text{if } x = 0^n, \\ \text{dp}(x - \chi(\{e(x, S)\}), S), & \text{if } e(x, S) > 0, \\ 1 + \min_{T \in \mathcal{S}, e(x, T) > 0} \{\text{dp}(x - \chi(\{e(x, T)\}), T)\}, & \text{if } e(x, S) = 0. \end{cases}$$

The new recursion also solves SMC_D , and the answer is equal to $\min_{S \in \mathcal{S}} \{\text{dp}(D^n, S)\}$.

Examine the underlying transition graph between pairs (x, S) . We can see that there is a transition between two pairs (x, S) and (y, T) only if $y_i = x_i + 1$ for exactly one i , and $y_i = x_i$ for other i . This is the n -dimensional lattice graph $Q(D, n)$. Thus we can apply our quantum algorithm with a few modifications:

- We now run Grover's search over (x, S) with fixed $|x|$ for all $S \in \mathcal{S}$. This adds a $\text{poly}(m, n)$ factor to each run of Grover's search.
- Since we are searching for the minimum value of dp , we actually need a quantum algorithm for finding the minimum instead of Grover's search. We can use the well-known quantum minimum finding algorithm that retains the same query complexity as Grover's search [14]³. It introduces only an additional $O(\log n)$ factor for the queries of minimum finding to encode the values of dp , since $\text{dp}(x, S)$ can be as large as Dn .
- A single query for a transition between pairs (x, S) and (y, T) in this case returns the value of the value added to the dp at transition, which is either 0 or 1. If these pairs are not connected in the transition graph, the query can return ∞ . Note that such query can be implemented in $\text{poly}(m, n)$ time.

Since the total number of runs of Grover's search is $O(K \log n)$, the additional factor incurred is $\text{poly}(m, n)^{O(K \log n)}$. This provides a quantum algorithm for this problem with total time complexity

$$\text{poly}(m, n)^{O(K \log n)} \cdot n^{O(D^2 K \log n)} T_D^n = m^{O(K \log n)} n^{O(D^2 K \log n)} T_D^n.$$

Therefore, we have the following result.

► **Theorem 8.** *Assuming the QRAM model of computation, there exists a quantum algorithm that solves SMC_D in time and space $\text{poly}(m, n)^{\log n} T_D^n$, where $T_D < D + 1$.*

7.2 Related problems

We are also aware of a couple of other works that implement the dynamic programming on the $\{0, 1, \dots, D\}^n$ n -dimensional lattice.

Psarftis examined the job scheduling problem [28], with application to aircraft landing scheduling. The problem requires ordering n groups of jobs with D identical jobs in each group. A cost transition function is given: the cost of processing a job belonging to group j after processing a job belonging to group i is given by $f(i, j, d_1, \dots, d_n)$, where d_i is the number of jobs left to process. The task is to find an ordering of the nD jobs that minimizes the total cost. This is almost exactly the setting for our quantum algorithm, hence we get $\text{poly}(n)^{\log n} T_D^n$ time quantum algorithm. Psarftis proposed a classical $O(n^2(D+1)^n)$ time

³ Note that this algorithm assumes queries with zero error, but we apply it to bounded-error queries. However, it consists of multiple runs of Grover's search, so we can still use the result of [23] to avoid the additional logarithmic factor.

dynamic programming algorithm. Note that if $f(i, j, d_1, \dots, d_n)$ are unstructured (can be arbitrary values), then there does not exist a faster classical algorithm by the lower bound of Section 3.

However, if $f(i, j, d_1, \dots, d_n)$ are structured or can be computed efficiently by an oracle, there exist more efficient classical algorithms for these kinds of problems. For instance, the many-visits travelling salesman problem (MV-TSP) asks for the shortest route in a weighted n -vertex graph that visits vertex i exactly D_i times. In this case, $f(i, j, d_1, \dots, d_n) = w(i, j)$, where $w(i, j)$ is the weight of the edge between i and j . The state-of-the-art classical algorithm by Kowalik et al. solves this problem in $\tilde{O}(4^n)$ time and space [32]. Thus, our quantum algorithm does not provide an advantage. It would be quite interesting to see if there exists a quantum speedup for this MV-TSP algorithm.

Lastly, Gromicho et al. proposed an exact algorithm for the job-shop scheduling problem [20, 31]. In this problem, there are n jobs to be processed on D machines. Each job consists of D tasks, with each task to be performed on a separate machine. The tasks for each job need to be processed in a specific order. The time to process job i on machine j is given by p_{ij} . Each machine can perform at most one task at any moment, but machines can perform the tasks in parallel. The problem is to schedule the starting times for all tasks so as to minimize the last ending time of the tasks. Gromicho et al. give a dynamic programming algorithm that solves the problem in time $O((p_{\max})^{2n}(D+1)^n)$, where $p_{\max} = \max_{i,j} \{p_{ij}\}$.

The states of their dynamic programming are also vectors in $\{0, 1, \dots, D\}^n$: a state x represents a partial completion of tasks, where x_i tasks of job i have already been completed. Their dynamic programming calculates the set of task schedulings for x that can be potentially extended to an optimal scheduling for all tasks. However, it is not clear how to apply Grover's search to calculate a whole set of schedulings. Therefore, even though the state space is the same as in our algorithm, we do not know whether it is possible to apply it in this case.

References

- 1 Amir Abboud. Fine-Grained Reductions and Quantum Speedups for Dynamic Programming. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:13, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.8.
- 2 Daniel S. Abrams and Seth Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Phys. Rev. Lett.*, 79:2586–2589, September 1997. doi:10.1103/PhysRevLett.79.2586.
- 3 Andris Ambainis. Quantum search with variable times. *Theory of Computing Systems*, 47(3):786–807, 2010. doi:10.1007/s00224-009-9219-1.
- 4 Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '19*, page 1783–1793, USA, 2019. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975482.107.
- 5 Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962. doi:10.1145/321105.321111.
- 6 Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. doi:10.1137/S0097539796300933.

- 7 Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. A note on exact algorithms for vertex ordering problems on graphs. *Theory of Computing Systems*, 50(3):420–432, 2012. doi:10.1007/s00224-011-9312-0.
- 8 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.
- 9 David Burshtein and Gadi Miller. Asymptotic enumeration methods for analyzing LDPC codes. *IEEE Transactions on Information Theory*, 50:1115–1131, 2004. doi:10.1109/TIT.2004.828064.
- 10 Mitchell Chiew, Kooper de Lacy, Chao-Hua Yu, Sam Marsh, and Jingbo B. Wang. Graph comparison via nonlinear quantum search. *Quantum Information Processing*, 18:302, August 2019. doi:10.1007/s11128-019-2407-2.
- 11 Arjan Cornelissen, Stacey Jeffery, Maris Ozols, and Alvaro Piedrafita. Span Programs and Quantum Time Complexity. In Javier Esparza and Daniel Král, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.26.
- 12 Marek Cygan and Marcin Pilipczuk. Faster exact bandwidth. In *Graph-Theoretic Concepts in Computer Science*, pages 101–109, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-92248-3_10.
- 13 N. G. de Bruijn, C^A. van Ebbenhorst Tengbergen, and D. Kruyswijk. On the set of divisors of a number. *Nieuw Archief voor Wiskunde*, 23(2):191–193, 1951. URL: <https://research.tue.nl/en/publications/on-the-set-of-divisors-of-a-number>.
- 14 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum, 1996. arXiv:quant-ph/9607014.
- 15 Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, USA, 1 edition, 2009. doi:10.1017/CB09780511801655.
- 16 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer Science & Business Media, 2010.
- 17 Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, 2008. doi:10.1103/PhysRevLett.100.160501.
- 18 Adam Glos, Martins Kokainis, Ryuhei Mori, and Jevgēnijs Vihrovs. Quantum speedups for dynamic programming on n -dimensional lattice graphs, 2021. arXiv:2104.14384.
- 19 I. J. Good. Saddle-point Methods for the Multinomial Distribution. *The Annals of Mathematical Statistics*, 28(4):861–881, 1957. doi:10.1214/aoms/1177706790.
- 20 Joaquim A. S. Gromicho, Jelke J. van Hoorn, Francisco Saldanha da Gama, and Gerrit T. Timmer. Solving the job-shop scheduling problem optimally by dynamic programming. *Computers & Operations Research*, 39(12):2968–2977, 2012. doi:10.1016/j.cor.2012.02.024.
- 21 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/237814.237866.
- 22 Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of SIAM*, 10(1):196–210, 1962. doi:10.1145/800029.808532.
- 23 Peter Høyer, Michele Mosca, and Ronald de Wolf. Quantum search on bounded-error inputs. In *Automata, Languages and Programming*, ICALP'03, page 291–299, Berlin, Heidelberg, 2003. Springer-Verlag. doi:10.1007/3-540-45061-0_25.
- 24 Qiang-Sheng Hua, Yuexuan Wang, Dongxiao Yu, and Francis C.M. Lau. Dynamic programming based algorithms for set multicover and multiset multicover problems. *Theoretical Computer Science*, 411(26):2467–2474, 2010. doi:10.1016/j.tcs.2010.02.016.

- 25 Bronisław Knaster, Casimir Kuratowski, and Stefan Mazurkiewicz. Ein beweis des fixpunktsatzes für n -dimensionale simplexe. *Fundamenta Mathematicae*, 14(1):132–137, 1929. doi:10.4064/fm-14-1-132-137.
- 26 Masayuki Miyamoto, Masakazu Iwamura, Koichi Kise, and François Le Gall. Quantum speedup for the minimum steiner tree problem. In Donghyun Kim, R. N. Uma, Zhipeng Cai, and Dong Hoon Lee, editors, *Computing and Combinatorics*, pages 234–245, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-58150-3_19.
- 27 Jesper Nederlof. Inclusion exclusion for hard problems. Master’s thesis, Utrecht University, 2008. URL: <https://webpace.science.uu.nl/~neder003/MScThesis.pdf>.
- 28 Harilaos N. Psaraftis. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research*, 28(6):1347–1359, 1980. doi:10.1287/opre.28.6.1347.
- 29 Kazuya Shimizu and Ryuhei Mori. Exponential-time quantum algorithms for graph coloring problems. In Yoshiharu Kohayakawa and Flávio Keidi Miyazawa, editors, *LATIN 2020: Theoretical Informatics*, pages 387–398, Cham, 2020. Springer International Publishing. arXiv:1907.00529.
- 30 Seiichiro Tani. Quantum Algorithm for Finding the Optimal Variable Ordering for Binary Decision Diagrams. In Susanne Albers, editor, *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*, volume 162 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SWAT.2020.36.
- 31 Jelke J. van Hoorn, Agustín Nogueira, Ignacio Ojea, and Joaquim A. S. Gromicho. An corrigendum on the paper: Solving the job-shop scheduling problem optimally by dynamic programming. *Computers & Operations Research*, 78:381, 2017. doi:10.1016/j.cor.2016.09.001.
- 32 Łukasz Kowalik, Shaohua Li, Wojciech Nadara, Marcin Smulewicz, and Magnus Wahlström. Many Visits TSP Revisited. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:22, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2020.66.

A Depth of recursion

Note that the algorithm stops the recursive calls if for at least one k , we have $W_k = W_{k+1}$, in which case it runs the classical dynamic programming on the whole sublattice at step 1. That happens when

$$\left| \sum_{d=1}^D \alpha_{k,d} dn_d \right| = \left| \sum_{d=1}^D \alpha_{k+1,d} dn_d \right|.$$

If this is true, then we also have $\sum_{d=1}^D \alpha_{k+1,d} dn_d - \sum_{d=1}^D \alpha_{k,d} dn_d = c$ for some constant $c < 1$. By regrouping the terms, we get

$$\sum_{d=1}^D (\alpha_{k+1,d} - \alpha_{k,d}) dn_d = c.$$

Denote $h := \min_{d \in [D]} \{\alpha_{k+1,d} - \alpha_{k,d}\}$. Then $\sum_{d=1}^D dn_d \leq \frac{c}{h}$. Note that the left hand side is the maximum total weight of a vertex. However, at each recursive call the difference between the vertices with the minimum and maximum total weights decreases twice, since the VTS call at step 4 runs over the vertices with weight half the current difference. Since c and h is constant, after $O(\log(nD)) = O(\log n)$ recursive calls the recursion stops. Moreover, the classical dynamic programming then runs on a sublattice of constant size, hence adds only a factor of $O(1)$ to the overall complexity.

Lastly, we can address the contribution of the constant factor of VTS from Theorem 1 to the complexity of our algorithm. At one level of recursion there are $K + 1$ nested applications of VTS, and there are $O(\log n)$ levels of recursion. Therefore, the total overhead incurred is $O(1)^{O(K \log n)} = \text{poly}(n)$, since K is a constant.

B Complexity technicalities of the optimization program

For correctness, we need to address the following two subtleties.

- The numbers $\alpha_k n$ might not be integer; in Algorithm 1, the weights of the layers are defined by $W_k = \lfloor \alpha_k n \rfloor$. This is a problem, since the inequalities in the program use precisely the numbers $\alpha_{k,d}$. Examine the coefficient $[x_1^{\lfloor \alpha_1 n \rfloor} \cdots x_m^{\lfloor \alpha_m n \rfloor}] p(x_1, \dots, x_m)$ in such general case (when we need to round the powers). Let $\delta_k := \alpha_k n - \lfloor \alpha_k n \rfloor$, here $0 \leq \delta_k < 1$. Then, by Theorem 2 (1),

$$[x_1^{\lfloor \alpha_1 n \rfloor} \cdots x_m^{\lfloor \alpha_m n \rfloor}] p(x_1, \dots, x_m)^n \leq \inf_{x_1, \dots, x_m \geq 0} \frac{p(x_1, \dots, x_m)^n}{x_1^{\alpha_1 n - \delta_1} \cdots x_m^{\alpha_m n - \delta_m}}. \quad (4)$$

Now let $\hat{x}_1, \dots, \hat{x}_m$ be the arguments that achieve $\inf_{x_1, \dots, x_m \geq 0} \frac{p(x_1, \dots, x_m)}{x_1^{\alpha_1} \cdots x_m^{\alpha_m}}$. Since $0 \leq \delta_k < 1$, we have $\hat{x}_k^{\delta_k} \leq \max\{\hat{x}_k, 1\}$. Hence, (4) is at most

$$(\hat{x}_1^{\delta_1} \cdots \hat{x}_m^{\delta_m}) \cdot \frac{p(\hat{x}_1, \dots, \hat{x}_m)^n}{\hat{x}_1^{\alpha_1 n} \cdots \hat{x}_m^{\alpha_m n}} \leq \left(\prod_{k=1}^m \max\{\hat{x}_k, 1\} \right) \cdot \left(\inf_{x_1, \dots, x_m \geq 0} \frac{p(x_1, \dots, x_m)}{x_1^{\alpha_1} \cdots x_m^{\alpha_m}} \right)^n.$$

As the additional factor is a constant, we can ignore it in the complexity.

- The second issue is when $W_k = W_{k+1}$ for some k . Then according to Algorithm 1, we run the classical algorithm with complexity $\tilde{\Theta}((D+1)^n)$. However, in that case n is constant (see Appendix A), which gives only a constant factor to the complexity.

C Proof of the optimality of the optimization program

- First, we prove that $\text{OPT}(D, K, \{\alpha_{k,d}\})$ has a feasible solution. For that, we need to show that all polynomials in the program can be upper bounded by a constant for some fixed values of the variables.

First of all, $\frac{P_d(x)}{x^{\alpha_{1,d}}}$ is upper bounded by $d + 1$ (setting $x = 1$). Now fix k and examine the values $\frac{S_{k,d}(x_{k,k}, \dots, x_{k,K+1})}{x_{k,k}^{\alpha_{k,d}} \cdots x_{k,K+1}^{\alpha_{K+1,d}}}$. Examine only such assignments of the variables $x_{k,j}$ that $x_{k,k} x_{k,k+1} = 1$ and $x_{k,j} = 1$ for all other $j > k + 1$. Now we write the polynomial as a univariate polynomial $S_{k,d}(y) := S_{k,d}(1/y, y, 1, \dots, 1)$. Note that for any summand of $S_{k,d}(y)$, if it contains some T_i^2 as a factor, then it is of the form $x_{k,k}^{p_k} x_{k,k+1}^{p_{k+1}} \cdot T_i^2 = y^i T_i^2$. Hence the polynomial can be written as $S_{k,d}(y) = \sum_{i=0}^d c_i y^i T_i^2$ for some constants c_1, \dots, c_d . From this we can rewrite the corresponding program inequality and express T_d^2 :

$$T_d^2 \geq \frac{\sum_{i=0}^d c_i y^i T_i^2}{y^{(\alpha_{k+1,d} - \alpha_{k,d})d}} \quad (5)$$

$$T_d^2 \geq \frac{\sum_{i=0}^{d-1} c_i y^i T_i^2}{y^{(\alpha_{k+1,d} - \alpha_{k,d})d}} + y^{(1 - \alpha_{k+1,d} + \alpha_{k,d})d} c_d T_d^2$$

$$T_d^2 \geq \frac{1}{1 - y^{(1 - \alpha_{k+1,d} + \alpha_{k,d})d} c_d} \cdot \frac{\sum_{i=0}^{d-1} c_i y^i T_i^2}{y^{(\alpha_{k+1,d} - \alpha_{k,d})d}}.$$

Note that c_d are constants that do not depend on T_i . If the right hand side is negative, then it follows that the original inequality Eq. (5) does not hold. Thus we need to pick such y that the right hand side is positive for all d . Hence we require that

$$y < \left(\frac{1}{c_d} \right)^{\frac{1}{(1-\alpha_{k+1,d}+\alpha_{k,d})^d}}.$$

Since the right hand side is a constant that does not depend on T_i , we can pick such y that satisfies this inequality for all d . Then it follows that all T_i is also upper bounded by some constants (by induction on i).

- Now the question remains whether the optimal solution to $\text{OPT}(D, K, \{\alpha_{k,d}\})$ gives the optimal complexity. That is, is the complexity $T_1^n \cdots T_D^{nD}$ given by the optimal solution of the optimization program such that T_D is the smallest possible?

Suppose that indeed the complexity of the algorithm is upper bounded by $T_1^n \cdots T_D^{nD}$ for some T_1, \dots, T_D . We will derive a corresponding feasible point for the optimization program.

Examine the complexity of the algorithm for $n_1 = b_1 n, \dots, n_D = b_D n$ for some fixed rational b_i such that $b_1 + \dots + b_D = 1$. The coefficients of the polynomials P and S_k give the complexity of the corresponding part of the algorithm (precalculation, and quantum search until the k -th level, respectively). Such coefficients are of the form $[x_1^{\alpha_1 n} \cdots x_m^{\alpha_m n}] \prod_{d=1}^D p_d(x_1, \dots, x_m)^{n_d}$. Let $A_d := T_d$, if $p = P$, and $A_d := T_d^2$, if $p = S_k$. Then we have

$$A_1^{n_1} \cdots A_D^{n_D} \geq [x_1^{\alpha_1 n} \cdots x_m^{\alpha_m n}] \prod_{d=1}^D p_d(x_1, \dots, x_m)^{n_d}. \quad (6)$$

On the other hand, (6) is at least

$$\Omega \left(\left(\inf_{x_1, \dots, x_m > 0} \prod_{d=1}^D \left(\frac{p_d(x_1, \dots, x_m)}{x_1^{a_{1,d}} \cdots x_m^{a_{m,d}}} \right)^{b_d} \right)^n \right)$$

when n grows large by Theorem 2 (2) (setting $a_{i,d} := \alpha_{i,d} d$). Then, in the limit $n \rightarrow \infty$, we have

$$A_1^{b_1} \cdots A_D^{b_D} \geq \inf_{x_1, \dots, x_m > 0} \prod_{d=1}^D \left(\frac{p_d(x_1, \dots, x_m)}{x_1^{a_{1,d}} \cdots x_m^{a_{m,d}}} \right)^{b_d}. \quad (7)$$

Now let Δ_{D-1} be the standard D -simplex defined by $\{b \in \mathbb{R}^D \mid b_1 + \dots + b_D = 1, b_d \geq 0\}$. Define $F_d(x) := \frac{p_d(x_1, \dots, x_m)}{x_1^{a_{1,d}} \cdots x_m^{a_{m,d}}}$, and $F(b, x) := \prod_{d=1}^D F_d(x)^{b_d}$ for $b \in \Delta_{D-1}$ and $x \in \mathbb{R}_{>0}^m$.

First, we prove that that for a fixed b , the function $F(b, x)$ is strictly convex. Examine the polynomial $p_d(x_1, \dots, x_m)$, which is either $P_d(x)$ or $S_{k,d}(x_{k,k}, \dots, x_{k,K+1})$. It was shown in [19], Theorem 6.3 that if the coefficients of $p_d(x_1, \dots, x_m)$ are non-negative, and the points (c_1, \dots, c_m) , at which

$$[x_1^{c_1} \cdots x_m^{c_m}] p_d(x_1, \dots, x_m) > 0,$$

linearly span an m -dimensional space, then $\log(F_d(x))$ is a strictly convex function. If $p_d = P_d$, then this property immediately follows, because there is just one variable x and the polynomial is non-constant. For $p_d = S_{k,d}$, the polynomial consists of summands of the form $T_{c_{k+1}-c_k}^2 x_{k,k}^{c_k} x_{k,k+1}^{c_{k+1}} \cdots x_{k,K+1}^{c_{K+1}}$, for $c_k \leq c_{k+1} \leq \dots \leq c_{K+1}$. Note that the

coefficient $T_{c_{k+1}-c_k}^2$ is positive. Thus the points $(c_k, \dots, c_{K+1}) = (0, \dots, 0, 1, \dots, 1)$ indeed linearly span a $(K - k + 2)$ -dimensional space. Therefore, $\log(F_d(x))$ is strictly convex. Then also the function $\sum_{d=1}^D b_d \log(F_d(x)) = \log(F(b, x))$ is strictly convex (for fixed b), as the sum of strictly convex functions is convex. Therefore, $F(b, x)$ is strictly convex as well.

Therefore, the argument $\hat{x}(b)$ achieving $\inf_{x \in \mathbb{R}_{>0}^m} F(b, x)$ is unique. Let $\hat{F}_d(b) := F_d(\hat{x}(b))$ and define D subsets of the simplex $C_d := \{b \in \Delta_{D-1} \mid \hat{F}_d(b) \leq A_d\}$. We will apply the following result for these sets:

► **Theorem 9** (Knaster-Kuratowski-Mazurkiewicz lemma [25]). *Let the vertices of Δ_{D-1} be labeled by integers from 1 to D . Let C_1, \dots, C_D be a family of closed sets such that for any $I \subseteq [D]$, the convex hull of the vertices labeled by I is covered by $\cup_{d \in I} C_d$. Then $\cap_{d \in [D]} C_d \neq \emptyset$.*

We check that the conditions of the lemma apply to our sets. First, note that $F(b, x)$ is continuous and strictly convex for a fixed b , hence $\hat{x}(b)$ is continuous and thus $\hat{F}_d(b)$ is continuous as well. Therefore, the “threshold” sets C_d are closed.

Secondly, let $I \subseteq [D]$ and examine a point b in the convex hull of the simplex vertices labeled by I . For such a point, we have $b_d = 0$ for all $d \notin I$. For the indices $d \in I$, for at least one we should have $\hat{F}_d(b) \leq A_d$, otherwise the inequality in Eq. (7) would be contradicted. Note that it was stated only for rational b , but since $\hat{F}_d(b)$ are continuous and any real number can be approximated with a rational number to arbitrary precision, the inequality also holds for real b . Thus indeed any such b is covered by $\cup_{d \in I} C_d$.

Therefore, we can apply the lemma and it follows that there exists a point $b \in \Delta_{D-1}$ such that $A_d \geq \hat{F}_d(b)$ for all $d \in [D]$. The corresponding point $\hat{x}(b)$ is a feasible point for the examined set of inequalities in the optimization program.

D Proof of the quantum speedup

Examine the algorithm with only $K = 1$; the optimal complexity for any $K > 1$ cannot be larger, as we can simulate K levels with $K + 1$ levels by setting $\alpha_{2,d} = \alpha_{1,d} + \epsilon$ for $\epsilon \rightarrow 0$ for all $d \in [D]$. For simplicity, denote $\alpha_d := \alpha_{1,d}$.

- Now examine the precalculation inequalities in $\text{OPT}(D, 1, \{\alpha_{1,d}\})$. For any values of $\alpha_{1,d}$, if we set $x = 1$, we have $\frac{P_d(x)}{x^{\alpha_d}} = \frac{\sum_{i=0}^d x^i}{x^{\alpha_d}} = d + 1$. The derivative is equal to

$$\left(\frac{\sum_{i=0}^d x^i}{x^{\alpha_d}} \right)' = \frac{x^{\alpha_d} \cdot \sum_{i=1}^d i x^{i-1} - \alpha_d x^{\alpha_d-1} \cdot \sum_{i=0}^d x^i}{x^{2\alpha_d}} = \frac{d(d+1)}{2} - \alpha_d d(d+1)$$

at point $x = 1$. Thus when $\alpha_d < \frac{1}{2}$, the derivative is positive. It means that for arbitrary $\alpha_d < \frac{1}{2}$, there exists some $x(d)$ such that $\frac{P_d(x)}{x^{\alpha_d}} < d + 1$, and $\frac{P_d(x)}{x^{\alpha_d}}$ monotonically grows on $x \in [x(d), 1]$. Thus, for arbitrary setting of $\{\alpha_d\}$ such that $\alpha_d < \frac{1}{2}$ for all $d \in [D]$, we can take $\hat{x} := \max_{d \in [D]} \{x(d)\}$ as the common parameter, in which case all $\frac{P_d(\hat{x})}{\hat{x}^{\alpha_d}} < d + 1$.

- Now examine the set of the quantum search inequalities. Let $y := x_{1,1}$ and $z := x_{1,2}$ for simplicity. Then such inequalities are given by

$$T_d^2 \geq S_{1,d}(y, z) = \frac{\sum_{i=0}^d T_i^2 \sum_{p=0}^{d-i} y^p z^{p+i}}{y^{\alpha_d} z^{d/2}}.$$

50:22 Quantum Speedups for Dynamic Programming on n -Dimensional Lattice Graphs

Now restrict the variables to condition $yz = 1$. In that case, the polynomial above simplifies to

$$S_{1,d}(z) := \frac{\sum_{i=0}^d T_i^2 \sum_{p=0}^{d-i} z^i}{y^{\frac{d}{2}+d(\alpha_d-\frac{1}{2})} z^{d/2}} = \left(\sum_{i=0}^d T_i^2 (d-i+1) z^i \right) \cdot z^{d(\alpha_d-\frac{1}{2})}.$$

We now find such values of z and $\alpha_1, \dots, \alpha_D$ so that $S_{1,d}(z) < (d+1)^2$ for all $d \in [D]$, where T_1, \dots, T_D are any values such that $T_d \leq d+1$ for all $d \in [D]$. Denote $\hat{S}_{1,d}(z)$ to be $S_{1,d}(z)$ with $T_d = d+1$ for all $d \in [D]$, then $\hat{S}_{1,d}(z) < (d+1)^2$ as well. Now let T_d be the maximum of $\frac{P_d(\hat{x})}{\hat{x}^{\alpha_d d}}$ from the previous bullet and $\hat{S}_{1,d}(z)$. Then, $T_d < d+1$, and we have both $T_d \geq \frac{P_d(\hat{x})}{\hat{x}^{\alpha_d d}}$ and $T_d^2 \geq \hat{S}_{1,d}(z) \geq S_{1,d}(z)$, since $S_{1,d}(z)$ cannot become larger when T_d decrease.

Now we show how to find such z and $\alpha_1, \dots, \alpha_D$. Examine the sum in the polynomial $\hat{S}_{1,d}(z)$

$$\sum_{i=0}^d (i+1)^2 (d-i+1) z^i = (d+1) + \sum_{i=1}^d (i+1)^2 (d-i+1) z^i.$$

Examine the second part of the sum. We can find a sufficiently small value of $z \in (0, 1)$ such that this part is smaller than any value $\epsilon > 0$ for all $d \in [D]$. Now, let $\alpha_d = \frac{1}{2} - \frac{c}{d}$ for some constant $c > 0$. Then

$$z^{d(\alpha_d-\frac{1}{2})} = z^{-c}$$

for all $d \in [D]$. Thus, the total value of the sum now is at most $(d+1+\epsilon)z^{-c}$. As $z^{-1} > 1$, take a sufficiently small value of c so that this value is at most $(d+1)^2$.

E Numerical results for $K = 1$

$D = 1$	$D = 2$	$D = 3$
$T_1 = 1.86793$	$T_1 = 1.87788$	$T_1 = 1.89454$
$x = 0.464808$	$T_2 = 2.76626$	$T_2 = 2.77944$
$x_{1,1} = 6.0606$	$x = 0.595073$	$T_3 = 3.68995$
$x_{1,2} = 0.104715$	$x_{1,1} = 5.74769$	$x = 0.684299$
$\alpha_{1,1} = 0.317317$	$x_{1,2} = 0.12725$	$x_{1,1} = 5.41613$
	$\alpha_{1,1} = 0.314447$	$x_{1,2} = 0.146775$
	$\alpha_{1,2} = 0.337219$	$\alpha_{1,1} = 0.310059$
		$\alpha_{1,2} = 0.336865$
		$\alpha_{1,3} = 0.351627$

$D = 4$ $T_1 = 1.91039$
 $T_2 = 2.80346$
 $T_3 = 3.7035$
 $T_4 = 4.63207$
 $x = 0.747046$
 $x_{1,1} = 5.11625$
 $x_{1,2} = 0.163892$
 $\alpha_{1,1} = 0.306472$
 $\alpha_{1,2} = 0.335557$
 $\alpha_{1,3} = 0.351929$
 $\alpha_{1,4} = 0.362866$ $D = 5$ $T_1 = 1.92386$
 $T_2 = 2.828$
 $T_3 = 3.72975$
 $T_4 = 4.64486$
 $T_5 = 5.58737$
 $x = 0.792588$
 $x_{1,1} = 4.8582$
 $x_{1,2} = 0.178964$
 $\alpha_{1,1} = 0.304026$
 $\alpha_{1,2} = 0.334429$
 $\alpha_{1,3} = 0.351624$
 $\alpha_{1,4} = 0.36331$
 $\alpha_{1,5} = 0.371992$ $D = 6$ $T_1 = 1.93495$
 $T_2 = 2.85009$
 $T_3 = 3.75806$
 $T_4 = 4.6709$
 $T_5 = 5.600$
 $T_6 = 6.55224$
 $x = 0.826544$
 $x_{1,1} = 4.63595$
 $x_{1,2} = 0.192435$
 $\alpha_{1,1} = 0.302631$
 $\alpha_{1,2} = 0.333786$
 $\alpha_{1,3} = 0.351339$
 $\alpha_{1,4} = 0.363364$
 $\alpha_{1,5} = 0.372425$
 $\alpha_{1,6} = 0.379599$

A Note on the Join of Varieties of Monoids with \mathbf{LI}

Nathan Grosshans   

Fachbereich Elektrotechnik/Informatik, University of Kassel, Germany

Abstract

In this note, we give a characterisation in terms of identities of the join of \mathbf{V} with the variety of finite locally trivial semigroups \mathbf{LI} for several well-known varieties of finite monoids \mathbf{V} by using classical algebraic-automata-theoretic techniques. To achieve this, we use the new notion of essentially- \mathbf{V} stamps defined by Grosshans, McKenzie and Segoufin and show that it actually coincides with the join of \mathbf{V} and \mathbf{LI} precisely when some natural condition on the variety of languages corresponding to \mathbf{V} is verified.

This work is a kind of rediscovery of the work of J. C. Costa around 20 years ago from a rather different angle, since Costa's work relies on the use of advanced developments in profinite topology, whereas what is presented here essentially uses an algebraic, language-based approach.

2012 ACM Subject Classification Theory of computation \rightarrow Formal languages and automata theory; Theory of computation \rightarrow Algebraic language theory

Keywords and phrases Varieties of monoids, join, \mathbf{LI}

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.51

Acknowledgements I want to thank Thomas Place, who suggested the link between essentially- \mathbf{V} stamps and $\mathbf{V} \vee \mathbf{LI}$, but also Luc Segoufin who started the discussion with Thomas Place and encouraged me to write the present article. My thanks go as well to the anonymous referees for their helpful comments and suggestions. Finally, I want to mention that the introductions of Jean-Éric Pin's future book on algebraic automata theory and of Marc Zeitoun's works cited in the references have been helpful inspirations for my own introduction.

1 Introduction

One of the most fundamental problems in finite automata theory is the one of *characterisation*: given some subclass of the class of regular languages, find out whether there is a way to characterise those languages using some class of finite objects. This problem is often linked to and motivated by the problem of *decidability*: given some subclass of the class of regular languages, find out whether there exists an algorithm testing the membership of any regular language in that subclass. The obvious approach to try to find a characterisation of a class of regular languages would be to look for properties shared by all the minimal finite automata of those languages. If we find such characterising properties, we can then ask whether they can be checked by an algorithm to answer the problem of decidability for this class of languages. However, one of the most fruitful approaches of those two problems has been the *algebraic approach*, in which we basically replace automata with morphisms into monoids: a language L over an alphabet Σ is then said to be recognised by a morphism φ into a monoid M if and only if L is the inverse image by φ of a subset of M . Under this notion of recognition, each language has a minimal morphism recognising it, the *syntactic morphism* into the *syntactic monoid* of that language, that are minimal under some notion of division. The fundamental result on which this algebraic approach relies is that a language is regular if and only if its syntactic monoid is finite. One can thus try to find a characterisation of some class of regular languages by looking at the algebraic properties of the syntactic monoids of these languages.

And many such characterisations that are decidable were indeed successfully obtained since Schützenberger's seminal work in 1965 [18]. His famous result, that really started the field of *algebraic automata theory*, states that the star-free regular languages are exactly



© Nathan Grosshans;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 51; pp. 51:1–51:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

those whose syntactic monoids are finite and aperiodic. Another important early result in that vein is the one of Simon [19] characterising the piecewise testable languages as exactly those having a finite \mathfrak{J} -trivial syntactic monoid. Eilenberg [12] was the first to prove that such algebraic characterisations actually come as specific instances of a general bijective correspondence between varieties of finite monoids and varieties of languages – classes of, respectively, finite monoids and regular languages closed under natural operations. Thus, a class of regular languages can indeed be characterised by the syntactic monoids of these languages, as soon as it verifies some nice closure properties. Eilenberg’s result was later completed by Reiterman’s theorem [17], that uses a notion of identities defined using profinite topology and states that a class of finite monoids is a variety of finite monoids if and only if it is defined by a set of profinite identities. Therefore, one can always characterise the variety of finite monoids associated to a variety of languages by a set of profinite identities and, additionally, this characterisation often leads to decidability, especially when this set is finite. A great deal of research works have been conducted to characterise varieties of finite monoids or semigroups by profinite identities (see the book of Almeida [3] for an overview; see also the book chapter by Pin [15] for more emphasis on the “language” part).

A kind of varieties of finite monoids or semigroups that has attracted many research efforts aiming for characterisations through identities are the varieties defined as the join of two other varieties. Given two varieties of finite monoids \mathbf{V} and \mathbf{W} , the *join of \mathbf{V} and \mathbf{W}* , denoted by $\mathbf{V} \vee \mathbf{W}$, is the least variety of finite monoids containing both \mathbf{V} and \mathbf{W} . One of the main motivations to try to understand $\mathbf{V} \vee \mathbf{W}$ is that the variety of languages corresponding to it by the Eilenberg correspondence, $\mathcal{L}(\mathbf{V} \vee \mathbf{W})$, is the one obtained by considering direct products of automata recognising languages from both $\mathcal{L}(\mathbf{V})$ and $\mathcal{L}(\mathbf{W})$, the varieties of languages corresponding to, respectively, \mathbf{V} and \mathbf{W} . This is a fundamental operation on automata, and while it is straightforward that $\mathcal{L}(\mathbf{V} \vee \mathbf{W})$ is simply the least variety of languages containing both $\mathcal{L}(\mathbf{V})$ and $\mathcal{L}(\mathbf{W})$, this does not at all furnish a decidable characterisation of $\mathcal{L}(\mathbf{V} \vee \mathbf{W})$, let alone a set of identities defining $\mathbf{V} \vee \mathbf{W}$. Generally speaking, the problem of finding a set of identities defining $\mathbf{V} \vee \mathbf{W}$ is difficult (see [3, 23]): in fact, there exist two varieties of finite semigroups that have a decidable membership problem but whose join has an undecidable membership problem [1]. However, sets of identities have been found for many specific joins: have a look at [2, 4, 6, 22, 21, 7, 9, 10] for some examples.

In this paper, we give a general method to find a set of identities defining the join of an arbitrary variety of finite monoids \mathbf{V} and the *variety of finite locally trivial semigroups* \mathbf{LI} , as soon as one has a set of identities defining \mathbf{V} and \mathbf{V} verifies some criterion. Joins of that sort have been studied quite a lot in the literature we mentioned in the previous paragraph (e.g. in [6, 21, 9, 10]), but while these works usually rely heavily on profinite topology with some in-depth understanding of the structure of the elements of the so-called free pro- \mathbf{V} monoids and free pro- \mathbf{LI} semigroups, we present a method that reduces the use of profinite topology to the minimum and that relies mainly on algebraic and language-theoretic techniques. The variety \mathbf{LI} is well-known to correspond to the class of languages for which membership only depends on bounded-length prefixes and suffixes of words. In [13], McKenzie, Segoufin and the author introduced the notion of *essentially- \mathbf{V}* stamps (surjective morphisms $\varphi: \Sigma^* \rightarrow M$ for Σ an alphabet and M a finite monoid) to characterise the built-in ability that programs over monoids in \mathbf{V} have to treat separately some constant-length beginning and ending of a word. Informally said, a stamp is *essentially- \mathbf{V}* when it behaves like a stamp into a monoid of \mathbf{V} as soon as a sufficiently long beginning and ending of the input word has been fixed. Our method builds on two results, that we prove in this article.

1. The first result is a characterisation in terms of identities of the class **EV** of essentially-**V** stamps given a set of identities E defining **V**: a stamp is in **EV** if and only if it satisfies all identities $x^\omega yuzt^\omega = x^\omega yvzt^\omega$ for $u = v$ an identity in E and where x, y, z, t do appear neither in u nor in v .
2. The second result says that **EV** and $\mathbf{V} \vee \mathbf{LI}$ do coincide if and only if **V** verifies some criterion, that can be formulated in terms of quotient-expressibility in $\mathcal{L}(\mathbf{V})$: any language $L \in \mathcal{L}(\mathbf{V})$ must, for an arbitrary choice of x, y , be such that the quotient $u^{-1}Lv^{-1}$ for u and v long enough can be expressed as the quotient $(xu)^{-1}K(vy)^{-1}$ for a $K \in \mathcal{L}(\mathbf{V})$.

Using these results, we can find a set of identities defining $\mathbf{V} \vee \mathbf{LI}$ as soon as a set of identities defining **V** is known by proving that **V** verifies the criterion in point 2. Note that actually, for technical reasons, we work with the so-called *ne*-variety of stamps corresponding to $\mathbf{V} \vee \mathbf{LI}$ rather than directly with the variety of finite semigroups $\mathbf{V} \vee \mathbf{LI}$, but this is not a problem since a variety of finite semigroups can always be seen as an *ne*-variety of stamps and vice versa. We apply this method to reprove characterisations of the join of **LI** with each of the well-known varieties of finite monoids **R**, **L**, **J** and any variety of finite groups.

The author noticed after proving those results that his work actually forms a kind of rediscovery of the work of J. C. Costa in [9]. He defines an operator U associating to each set of identities E the exact same new set $U(E)$ of identities as in point 1. Costa then defines a property of cancellation for varieties of finite semigroups such that for any **V** verifying it, $U(E)$ defines $\mathbf{V} \vee \mathbf{LI}$ for E defining **V**. He finally uses this result to derive characterisations of $\mathbf{V} \vee \mathbf{LI}$ for all the cases we are treating in our paper and many more.

What is, then, the contribution of our article? In a nutshell, it does mainly use algebraic and language-theoretic techniques while Costa's work relies heavily on profinite topology. In our setting, once the stage is set, all proofs are quite straightforward without real difficulties and rely on classical language-theoretic characterisations of the varieties under consideration. This is to contrast with Costa's work, that for instance draws upon the difficult analysis of the elements of free pro-**R** monoids by Almeida and Weil [5] to characterise $\mathbf{R} \vee \mathbf{LI}$.

Organisation of the article. Section 2 is dedicated to the necessary preliminaries. In Section 3, we recall the definition of essentially-**V** stamps and prove the characterisation by identities of point 1 above. Section 4 is then dedicated to the necessary and sufficient criterion for **EV** and $\mathbf{V} \vee \mathbf{LI}$ to coincide presented in point 2 and finally those results are applied to specific cases in Section 5. We finish with a short conclusion.

2 Preliminaries

We briefly introduce the mathematical material used in this paper. For the basics and the classical results of automata theory, we refer the reader to the two classical references of the domain by Eilenberg [11, 12] and Pin [14]. For definitions and results specific to varieties of stamps and associated profinite identities, see the articles by Straubing [20] and by Pin and Straubing [16]. We also assume some basic knowledge of topology.

General notations. Let $i \in \mathbb{N}$ be a natural number. We shall denote by $[i]$ the set of all natural numbers $n \in \mathbb{N}$ verifying $1 \leq n \leq i$.

Words and languages. Let Σ be a finite alphabet. We denote by Σ^* the set of all finite words over Σ . We also denote by Σ^+ the set of all finite non empty words over Σ , the empty word being denoted by ε . Our alphabets and words will always be finite, without further

mention of this fact. Given a word $w \in \Sigma^*$, we denote its length by $|w|$ and the set of letters it contains by $\text{alph}(w)$. Given $n \in \mathbb{N}$, we denote by $\Sigma^{\geq n}$, Σ^n and $\Sigma^{< n}$ the set of words over Σ of length, respectively, at least n , exactly n and less than n .

A *language over Σ* is a subset of Σ^* . A language is *regular* if it is recognised by a deterministic finite automaton. The *quotient of a language L over Σ relative to the words u and v over Σ* is the language, denoted by $u^{-1}Lv^{-1}$, of the words w such that $u w v \in L$.

Monoids, semigroups and varieties. A *semigroup* is a non-empty set equipped with an associative law that we will write multiplicatively. A *monoid* is a semigroup with an identity. An example of a semigroup is Σ^+ , the free semigroup over Σ . Similarly Σ^* is the free monoid over Σ . A *morphism φ from a semigroup S to a semigroup T* is a function from S to T such that $\varphi(xy) = \varphi(x)\varphi(y)$ for all $x, y \in S$. A morphism of monoids additionally requires that the identity is preserved. A semigroup T is a *subsemigroup* of a semigroup S if T is a subset of S and is equipped with the restricted law of S . Additionally the notion of submonoids requires the presence of the identity. A semigroup T *divides* a semigroup S if T is the image by a semigroup morphism of a subsemigroup of S . Division of monoids is defined in the same way. The *Cartesian (or direct) product* of two semigroups is simply the semigroup given by the Cartesian product of the two underlying sets equipped with the Cartesian product of their laws. An element s of a semigroup is *idempotent* if $ss = s$.

A *variety of finite monoids* is a non-empty class of finite monoids closed under Cartesian product and monoid division. A *variety of finite semigroups* is defined similarly. When dealing with varieties, we consider only finite monoids and semigroups, so we will drop the adjective finite when talking about varieties in the rest of this article.

Varieties of stamps. Let $f: \Sigma^* \rightarrow \Gamma^*$ be a morphism from the free monoid over an alphabet Σ to the free monoid over an alphabet Γ , that we might call an *all-morphism*. We say that f is an *ne-morphism* (non-erasing morphism) whenever $f(\Sigma) \subseteq \Gamma^+$.

We call *stamp* a surjective morphism $\varphi: \Sigma^* \rightarrow M$ for Σ an alphabet and M a finite monoid. We say that a stamp $\varphi: \Sigma^* \rightarrow M$ *all-divides* (respectively *ne-divides*) a stamp $\psi: \Gamma^* \rightarrow N$ whenever there exists an *all-morphism* (respectively *ne-morphism*) $f: \Sigma^* \rightarrow \Gamma^*$ and a surjective morphism $\alpha: \mathfrak{Im}(\psi \circ f) \rightarrow M$ such that $\varphi = \alpha \circ \psi \circ f$. The *direct product* of two stamps $\varphi: \Sigma^* \rightarrow M$ and $\psi: \Sigma^* \rightarrow N$ is the stamp $\varphi \times \psi: \Sigma^* \rightarrow K$ such that K is the submonoid of $M \times N$ generated by $\{(\varphi(a), \psi(a)) \mid a \in \Sigma\}$ and $\varphi \times \psi(a) = (\varphi(a), \psi(a))$ for all $a \in \Sigma$.

An *all-variety of stamps* (respectively *ne-variety of stamps*) is a non-empty class of stamps closed under direct product and *all-division* (respectively *ne-division*).

We will often use the following characteristic index of stamps, defined in [8]. Consider a stamp $\varphi: \Sigma^* \rightarrow M$. As M is finite there is a $k \in \mathbb{N}_{>0}$ such that $\varphi(\Sigma^{2k}) = \varphi(\Sigma^k)$: this implies that $\varphi(\Sigma^k)$ is a semigroup. The least such k is called the *stability index* of φ .

Varieties of languages. A language L over an alphabet Σ is *recognised by a monoid M* if there is a morphism $\varphi: \Sigma^* \rightarrow M$ and $F \subseteq M$ such that $L = \varphi^{-1}(F)$. We also say that φ *recognises L* . It is well known that a language is regular if and only if it is recognised by a finite monoid. The *syntactic congruence* of L , denoted by \sim_L , is the equivalence relation on Σ^* defined by $u \sim_L v$ for $u, v \in \Sigma^*$ whenever for all $x, y \in \Sigma^*$, $xuy \in L$ if and only if $xvy \in L$. The quotient Σ^*/\sim_L is a monoid, called *the syntactic monoid of L* , that recognises L via the *syntactic morphism η_L of L* sending any word u to its equivalence class $[u]_{\sim_L}$ for \sim_L . A stamp $\varphi: \Sigma^* \rightarrow M$ recognises L if and only if there exists a surjective morphism $\alpha: M \rightarrow \Sigma^*/\sim_L$ verifying $\eta_L = \alpha \circ \varphi$.

A class of languages \mathcal{C} is a correspondence that associates a set $\mathcal{C}(\Sigma)$ to each alphabet Σ . A (all-)variety of languages (respectively an *ne-variety of languages*) \mathcal{V} is a non-empty class of regular languages closed under Boolean operations, quotients and inverses of *all*-morphisms (respectively *ne*-morphisms). A classical result of Eilenberg [12, Chapter VII, Section 3] says that there is a bijective correspondence between varieties of monoids and varieties of languages: to each variety of monoids \mathbf{V} we can bijectively associate $\mathcal{L}(\mathbf{V})$ the variety of languages whose syntactic monoids belong to \mathbf{V} . This was generalised by Straubing [20] to varieties of stamps: to each *all*-variety (respectively *ne*-variety) of stamps \mathbf{V} we can bijectively associate $\mathcal{L}(\mathbf{V})$ the *all*-variety (respectively *ne*-variety) of languages whose syntactic morphisms belong to \mathbf{V} . Given two *all*-varieties (respectively *ne*-varieties) of stamps \mathbf{V}_1 and \mathbf{V}_2 , we have $\mathbf{V}_1 \subseteq \mathbf{V}_2 \Leftrightarrow \mathcal{L}(\mathbf{V}_1) \subseteq \mathcal{L}(\mathbf{V}_2)$.

For \mathbf{V} a variety of monoids, we define $\langle \mathbf{V} \rangle_{all}$ the *all*-variety of all stamps $\varphi: \Sigma^* \rightarrow M$ such that $M \in \mathbf{V}$. Of course, in that case $\mathcal{L}(\mathbf{V}) = \mathcal{L}(\langle \mathbf{V} \rangle_{all})$. Similarly, for \mathbf{V} a variety of semigroups, we define $\langle \mathbf{V} \rangle_{ne}$ the *ne*-variety of all stamps $\varphi: \Sigma^* \rightarrow M$ such that $\varphi(\Sigma^+) \in \mathbf{V}$. In that case, we consider $\mathcal{L}(\mathbf{V})$ to be the *ne*-variety of languages corresponding to $\langle \mathbf{V} \rangle_{ne}$. The operations $\langle \cdot \rangle_{all}$ and $\langle \cdot \rangle_{ne}$ form bijective correspondences between varieties of monoids and *all*-varieties of stamps and between varieties of semigroups and *ne*-varieties of stamps, respectively (see [20]).

Identities. Let Σ be an alphabet. Given $u, v \in \Sigma^*$, we set

$$r(u, v) = \min\{|M| \mid \exists \varphi: \Sigma^* \rightarrow M \text{ stamp s.t. } \varphi(u) \neq \varphi(v)\}$$

and $d(u, v) = 2^{-r(u, v)}$, using the conventions that $\min \emptyset = +\infty$ and $2^{-\infty} = 0$. Then d is a metric on Σ^* . The completion of the metric space (Σ^*, d) , denoted by $(\widehat{\Sigma^*}, \widehat{d})$, is a metric monoid called the *free profinite monoid on Σ^** . Its elements are all the formal limits $\lim_{n \rightarrow \infty} x_n$ of Cauchy sequences $(x_n)_{n \geq 0}$ in (Σ^*, d) and the metric d on Σ^* extends to a metric \widehat{d} on $\widehat{\Sigma^*}$ defined by $\widehat{d}(\lim_{n \rightarrow \infty} x_n, \lim_{n \rightarrow \infty} y_n) = \lim_{n \rightarrow \infty} d(x_n, y_n)$ for Cauchy sequences $(x_n)_{n \geq 0}$ and $(y_n)_{n \geq 0}$ in (Σ^*, d) . Note that, when it is clear from the context, we usually do not make the metric explicit when talking about a metric space. One important example of elements of $\widehat{\Sigma^*}$ is given by the elements $x^\omega = \lim_{n \rightarrow \infty} x^{n!}$ for all $x \in \Sigma^*$.

Every finite monoid M is considered to be a complete metric space equipped with the discrete metric d defined by $d(m, n) = \begin{cases} 0 & \text{if } m = n \\ 1 & \text{otherwise} \end{cases}$ for all $m, n \in M$. Every stamp

$\varphi: \Sigma^* \rightarrow M$ extends uniquely to a uniformly continuous morphism $\widehat{\varphi}: \widehat{\Sigma^*} \rightarrow M$ with $\widehat{\varphi}(\lim_{n \rightarrow \infty} x_n) = \lim_{n \rightarrow \infty} \varphi(x_n)$ for every Cauchy sequence $(x_n)_{n \geq 0}$ in Σ^* . Similarly, every *all*-morphism $f: \Sigma^* \rightarrow \Gamma^*$ extends uniquely to a uniformly continuous morphism $\widehat{f}: \widehat{\Sigma^*} \rightarrow \widehat{\Gamma^*}$ with $\widehat{f}(\lim_{n \rightarrow \infty} x_n) = \lim_{n \rightarrow \infty} f(x_n)$ for every Cauchy sequence $(x_n)_{n \geq 0}$ in Σ^* .

For $u, v \in \widehat{A^*}$ with A an alphabet, we say that a stamp $\varphi: \Sigma^* \rightarrow M$ *all-satisfies* (respectively *ne-satisfies*) the identity $u = v$ if for every *all*-morphism (respectively *ne*-morphism) $f: A^* \rightarrow \Sigma^*$, it holds that $\widehat{\varphi} \circ \widehat{f}(u) = \widehat{\varphi} \circ \widehat{f}(v)$. Given a set of identities E , we denote by $\llbracket E \rrbracket_{all}$ (respectively $\llbracket E \rrbracket_{ne}$) the class of stamps *all*-satisfying (respectively *ne*-satisfying) all the identities of E . When $\llbracket E \rrbracket_{all}$ (respectively $\llbracket E \rrbracket_{ne}$) is equal to an *all*-variety (respectively *ne*-variety) of stamps \mathbf{V} , we say that E *all-defines* (respectively *ne-defines*) \mathbf{V} .

► **Theorem 1** ([16, Theorem 2.1]). *A class of stamps is an all-variety (respectively ne-variety) of stamps if and only if it can be all-defined (respectively ne-defined) by a set of identities.*

To give some examples, the classical varieties of monoids **J**, **R** and **L** can be characterised by identities in the following way:

$$\begin{aligned} \langle \mathbf{R} \rangle_{all} &= \llbracket (ab)^\omega a = (ab)^\omega \rrbracket_{all} = \llbracket (ab)^\omega a = (ab)^\omega \rrbracket_{ne} \\ \langle \mathbf{L} \rangle_{all} &= \llbracket b(ab)^\omega = (ab)^\omega \rrbracket_{all} = \llbracket b(ab)^\omega = (ab)^\omega \rrbracket_{ne} \\ \langle \mathbf{J} \rangle_{all} &= \llbracket (ab)^\omega a = (ab)^\omega, b(ab)^\omega = (ab)^\omega \rrbracket_{all} = \llbracket (ab)^\omega a = (ab)^\omega, b(ab)^\omega = (ab)^\omega \rrbracket_{ne} . \end{aligned}$$

Finite locally trivial semigroups and the join operation. The variety **LI** of finite locally trivial semigroups is well-known to verify $\langle \mathbf{LI} \rangle_{ne} = \llbracket x^\omega y x^\omega = x^\omega \rrbracket_{ne}$ and to be such that for any alphabet Σ , the set $\mathcal{L}(\mathbf{LI})(\Sigma)$ consists of all Boolean combinations of languages of the form $u\Sigma^*$ or Σ^*u for $u \in \Sigma^*$, or equivalently of all languages of the form $U\Sigma^*V \cup W$ with $U, V, W \subseteq \Sigma^*$ finite (see [14, p. 38]).

Given a variety of monoids **V**, the join of **V** and **LI**, denoted by $\mathbf{V} \vee \mathbf{LI}$, is the inclusion-wise least variety of semigroups containing both **V** and **LI**. In fact, a finite semigroup S belongs to $\mathbf{V} \vee \mathbf{LI}$ if and only if there exist $M \in \mathbf{V}$ and $T \in \mathbf{LI}$ such that S divides the semigroup $M \times T$. (See [12, Chapter V, Exercise 1.1].) We can prove the following adaptation to *ne*-varieties of the classical results about joins (see the appendix for the proof).

► **Proposition 2.** *Let **V** be a variety of monoids. Then $\langle \mathbf{V} \vee \mathbf{LI} \rangle_{ne}$ is the inclusion-wise least *ne*-variety of stamps containing both $\langle \mathbf{V} \rangle_{all}$ and $\langle \mathbf{LI} \rangle_{ne}$. Moreover, $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})$ is the inclusion-wise least *ne*-variety of languages containing both $\mathcal{L}(\mathbf{V})$ and $\mathcal{L}(\mathbf{LI})$ and verifies that $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})(\Sigma)$ is the Boolean closure of $\mathcal{L}(\mathbf{V})(\Sigma) \cup \mathcal{L}(\mathbf{LI})(\Sigma)$ for each alphabet Σ .*

3 Essentially-**V** stamps

In this section, we give a characterisation of essentially-**V** stamps (first defined in [13]), for **V** a variety of monoids, in terms of identities. We first recall the definition.

► **Definition 3.** *Let **V** be a variety of monoids. Let $\varphi: \Sigma^* \rightarrow M$ be a stamp and let s be its stability index.*

*We say that φ is essentially-**V** whenever there exists a stamp $\mu: \Sigma^* \rightarrow N$ with $N \in \mathbf{V}$ such that for all $u, v \in \Sigma^*$, we have*

$$\mu(u) = \mu(v) \Rightarrow (\varphi(xuy) = \varphi(xvy) \quad \forall x, y \in \Sigma^s) .$$

*We will denote by **EV** the class of all essentially-**V** stamps.*¹

Now, we give a characterisation for a stamp to be essentially-**V**, based on a specific congruence depending on that stamp.

¹ Essentially-**V** stamps are called that way by analogy with quasi-**V** stamps and the class of essentially-**V** stamps is denoted by **EV** by analogy with **QV**, the notation for the class of quasi-**V** stamps. This makes sense since the initial motivation for the definition of essentially-**V** stamps was to capture the class of stamps into monoids of **V** that have the additional ability to treat separately some constant-length beginning and ending of a word. This ability can indeed be seen as orthogonal to the additional ability of stamps into monoids in **V** to perform modular counting on the positions of letters in a word, which is often handled by considering quasi-**V** stamps. (See [13] for more.) Our definition of **EV** does unfortunately not coincide with the usual definition of **EV**, that classically denotes the variety of monoids M such that the submonoid generated by the idempotents of M is in **V**. (This comes, among others, from the fact that the obtained variety of monoids does always contain at least all finite groups.)

► **Definition 4.** Let $\varphi: \Sigma^* \rightarrow M$ be a stamp and let s be its stability index. We define the equivalence relation \equiv_φ on Σ^* by $u \equiv_\varphi v$ for $u, v \in \Sigma^*$ whenever $\varphi(xuy) = \varphi(xvy)$ for all $x, y \in \Sigma^{\geq s}$.

► **Proposition 5.** Let $\varphi: \Sigma^* \rightarrow M$ be a stamp. Then \equiv_φ is a congruence of finite index and for any variety of monoids \mathbf{V} , we have $\varphi \in \mathbf{EV}$ if and only if $\Sigma^*/\equiv_\varphi \in \mathbf{V}$.

Proof. Let us denote by s the stability index of φ .

The equivalence relation \equiv_φ is a congruence because given $u, v \in \Sigma^*$ verifying $u \equiv_\varphi v$, for all $\alpha, \beta \in \Sigma^*$, we have $\alpha u \beta \equiv_\varphi \alpha v \beta$ since for any $x, y \in \Sigma^{\geq s}$, it holds that $\varphi(x\alpha u \beta y) = \varphi(x\alpha v \beta y)$ because $x\alpha, \beta y \in \Sigma^{\geq s}$. Furthermore, this congruence is of finite index because for all $u, v \in \Sigma^*$, we have that $\varphi(u) = \varphi(v)$ implies $u \equiv_\varphi v$.

Let now \mathbf{V} be a variety of monoids. Assume first that $\Sigma^*/\equiv_\varphi \in \mathbf{V}$. It is quite direct to see that $\varphi \in \mathbf{EV}$, as the stamp $\mu: \Sigma^* \rightarrow \Sigma^*/\equiv_\varphi$ defined by $\mu(w) = [w]_{\equiv_\varphi}$ for all $w \in \Sigma^*$ witnesses this fact. Assume then that $\varphi \in \mathbf{EV}$. This means that there exists a stamp $\mu: \Sigma^* \rightarrow N$ with $N \in \mathbf{V}$ such that for all $u, v \in \Sigma^*$, we have

$$\mu(u) = \mu(v) \Rightarrow (\varphi(xuy) = \varphi(xvy) \quad \forall x, y \in \Sigma^s) .$$

Now consider $u, v \in \Sigma^*$ such that $\mu(u) = \mu(v)$. For any $x, y \in \Sigma^{\geq s}$, we have that $x = x_1 x_2$ with $x_1 \in \Sigma^*$ and $x_2 \in \Sigma^s$ as well as $y = y_1 y_2$ with $y_1 \in \Sigma^s$ and $y_2 \in \Sigma^*$, so that $\varphi(xuy) = \varphi(x_1)\varphi(x_2 u y_1)\varphi(y_2) = \varphi(x_1)\varphi(x_2 v y_1)\varphi(y_2) = \varphi(xvy)$. Hence, $u \equiv_\varphi v$. Therefore, for all $u, v \in \Sigma^*$, we have that $\mu(u) = \mu(v)$ implies $u \equiv_\varphi v$, so we can define the mapping $\alpha: N \rightarrow \Sigma^*/\equiv_\varphi$ such that $\alpha(\mu(w)) = [w]_{\equiv_\varphi}$ for all $w \in \Sigma^*$. It is easy to check that α is actually a surjective morphism. Thus, we can conclude that Σ^*/\equiv_φ , which divides N , belongs to \mathbf{V} . ◀

Using this characterisation, we prove that given a set of identities ne -defining $\langle \mathbf{V} \rangle_{all}$ for a variety of monoids \mathbf{V} , we get a set of identities ne -defining \mathbf{EV} .

► **Proposition 6.** Let \mathbf{V} be a variety of monoids and let E be a set of identities such that $\langle \mathbf{V} \rangle_{all} = \llbracket E \rrbracket_{ne}$. Then \mathbf{EV} is an ne -variety of stamps and

$$\mathbf{EV} = \llbracket x^\omega y u z t^\omega = x^\omega y v z t^\omega \mid u = v \in E, x, y, z, t \notin \text{alph}(u) \cup \text{alph}(v) \rrbracket_{ne} .$$

Proof. Let

$$F = \{x^\omega y u z t^\omega = x^\omega y v z t^\omega \mid u = v \in E, x, y, z, t \notin \text{alph}(u) \cup \text{alph}(v)\} .$$

Central to the proof is the following claim.

▷ **Claim 7.** Let $\varphi: \Sigma^* \rightarrow M$ be a stamp. Consider the stamp $\mu: \Sigma^* \rightarrow \Sigma^*/\equiv_\varphi$ defined by $\mu(w) = [w]_{\equiv_\varphi}$ for all $w \in \Sigma^*$. It holds that for all $u, v \in \widehat{\Sigma}^*$,

$$\widehat{\mu}(u) = \widehat{\mu}(v) \Leftrightarrow (\widehat{\varphi}(\alpha^\omega \beta u \gamma \delta^\omega) = \widehat{\varphi}(\alpha^\omega \beta v \gamma \delta^\omega) \quad \forall \alpha, \beta, \gamma, \delta \in \Sigma^+) .$$

Before we prove Claim 7, we use it to prove that $\mathbf{EV} = \llbracket F \rrbracket_{ne}$.

Inclusion from left to right. Let $\varphi: \Sigma^* \rightarrow M$ be a stamp in \mathbf{EV} . Consider the stamp $\mu: \Sigma^* \rightarrow \Sigma^*/\equiv_\varphi$ defined by $\mu(w) = [w]_{\equiv_\varphi}$ for all $w \in \Sigma^*$. Since $\varphi \in \mathbf{EV}$, Proposition 5 tells us that $\Sigma^*/\equiv_\varphi \in \mathbf{V}$, hence $\mu \in \langle \mathbf{V} \rangle_{all}$.

Let us consider any identity $x^\omega y u z t^\omega = x^\omega y v z t^\omega \in F$. It is written on an alphabet B that is the union of the alphabet A on which $u = v \in E$ is written and of $x, y, z, t \in B \setminus A$. Let $f: B^* \rightarrow \Sigma^*$ be an ne -morphism. Since $\mu \in \langle \mathbf{V} \rangle_{all}$, we have that μ ne -satisfies the

identity $u = v$, so that $\widehat{\mu}(\widehat{f}(u)) = \widehat{\mu}(\widehat{f}(v))$. Notice that we have that $\widehat{f}(x^\omega) = f(x)^\omega$ as well as $\widehat{f}(t^\omega) = f(t)^\omega$ and that $f(x), f(y), f(z), f(t) \in \Sigma^+$ because f is non-erasing. Therefore, we have

$$\begin{aligned} \widehat{\varphi}(\widehat{f}(x^\omega y u z t^\omega)) &= \widehat{\varphi}(f(x)^\omega f(y) \widehat{f}(u) f(z) f(t)^\omega) \\ &= \widehat{\varphi}(f(x)^\omega f(y) \widehat{f}(v) f(z) f(t)^\omega) \\ &= \widehat{\varphi}(\widehat{f}(x^\omega y v z t^\omega)) \end{aligned}$$

by Claim 7. As this holds for any ne -morphism $f: B^* \rightarrow \Sigma^*$, we can conclude that φ ne -satisfies the identity $x^\omega y u z t^\omega = x^\omega y v z t^\omega$.

This is true for any identity in F , so $\varphi \in \llbracket F \rrbracket_{ne}$. In conclusion, $\mathbf{EV} \subseteq \llbracket F \rrbracket_{ne}$.

Inclusion from right to left. Let $\varphi: \Sigma^* \rightarrow M$ be a stamp in $\llbracket F \rrbracket_{ne}$. Consider the stamp $\mu: \Sigma^* \rightarrow \Sigma^*/\equiv_\varphi$ defined by $\mu(w) = [w]_{\equiv_\varphi}$ for all $w \in \Sigma^*$. We are now going to show that $\mu \in \langle \mathbf{V} \rangle_{all}$.

Take any identity $u = v \in E$ written on an alphabet A . There exists an identity $x^\omega y u z t^\omega = x^\omega y v z t^\omega \in F$ written on an alphabet B such that $A \subseteq B$ and $x, y, z, t \in B \setminus A$. Let $f: A^* \rightarrow \Sigma^*$ be an ne -morphism.

Take any $\alpha, \beta, \gamma, \delta \in \Sigma^+$. Let us define the ne -morphism $g: B^* \rightarrow \Sigma^*$ as the unique one which extends f by letting $g(x) = \alpha$, $g(y) = \beta$, $g(z) = \gamma$ and $g(t) = \delta$. Observe in particular that $\widehat{g}(w) = \widehat{f}(w)$ for any $w \in \widehat{A^*}$ and that $\widehat{g}(x^\omega) = g(x)^\omega = \alpha^\omega$ as well as $\widehat{g}(t^\omega) = \delta^\omega$. Now, as φ ne -satisfies $x^\omega y u z t^\omega = x^\omega y v z t^\omega$, we have that

$$\widehat{\varphi}(\alpha^\omega \beta \widehat{f}(u) \gamma \delta^\omega) = \widehat{\varphi}(\widehat{g}(x^\omega y u z t^\omega)) = \widehat{\varphi}(\widehat{g}(x^\omega y v z t^\omega)) = \widehat{\varphi}(\alpha^\omega \beta \widehat{f}(v) \gamma \delta^\omega) .$$

Since this holds for any $\alpha, \beta, \gamma, \delta \in \Sigma^+$, by Claim 7, we have that $\widehat{\mu}(\widehat{f}(u)) = \widehat{\mu}(\widehat{f}(v))$.

Therefore, $\widehat{\mu}(\widehat{f}(u)) = \widehat{\mu}(\widehat{f}(v))$ for any ne -morphism $f: A^* \rightarrow \Sigma^*$, which means that μ ne -satisfies $u = v$.

Since this holds for any $u = v \in E$, we have that $\mu \in \langle \mathbf{V} \rangle_{all}$, which implies that $\Sigma^*/\equiv_\varphi \in \mathbf{V}$ and thus $\varphi \in \mathbf{EV}$ by Proposition 5. In conclusion, $\llbracket F \rrbracket_{ne} \subseteq \mathbf{EV}$.

The claim still needs to be proved.

Proof of Claim 7. Let $\varphi: \Sigma^* \rightarrow M$ be a stamp of stability index s . Consider the stamp $\mu: \Sigma^* \rightarrow \Sigma^*/\equiv_\varphi$ defined by $\mu(w) = [w]_{\equiv_\varphi}$ for all $w \in \Sigma^*$. We now want to show that for all $u, v \in \widehat{\Sigma^*}$,

$$\widehat{\mu}(u) = \widehat{\mu}(v) \Leftrightarrow (\widehat{\varphi}(\alpha^\omega \beta u \gamma \delta^\omega) = \widehat{\varphi}(\alpha^\omega \beta v \gamma \delta^\omega) \quad \forall \alpha, \beta, \gamma, \delta \in \Sigma^+) .$$

Let $u, v \in \widehat{\Sigma^*}$. There exist two Cauchy sequences $(u_n)_{n \geq 0}$ and $(v_n)_{n \geq 0}$ in Σ^* such that $u = \lim_{n \rightarrow \infty} u_n$ and $v = \lim_{n \rightarrow \infty} v_n$. As Σ^*/\equiv_φ and M are discrete, we have that all four Cauchy sequences $(\mu(u_n))_{n \geq 0}$, $(\varphi(u_n))_{n \geq 0}$, $(\mu(v_n))_{n \geq 0}$ and $(\varphi(v_n))_{n \geq 0}$ are ultimately constant. So there exists $k \in \mathbb{N}$ such that $\widehat{\mu}(u) = \mu(u_k)$, $\widehat{\varphi}(u) = \varphi(u_k)$, $\widehat{\mu}(v) = \mu(v_k)$ and $\widehat{\varphi}(v) = \varphi(v_k)$.

Assume first that $\widehat{\mu}(u) = \widehat{\mu}(v)$. Take any $\alpha, \beta, \gamma, \delta \in \Sigma^+$. Since M is discrete, both Cauchy sequences $(\varphi(\alpha^{n!}))_{n \geq 0}$ and $(\varphi(\delta^{n!}))_{n \geq 0}$ are ultimately constant. So there exists $l \in \mathbb{N}$ such that for all $m \in \mathbb{N}, m \geq l$, we have $\widehat{\varphi}(\alpha^\omega) = \varphi(\alpha^{m!})$ and $\widehat{\varphi}(\delta^\omega) = \varphi(\delta^{m!})$. Hence, taking $m \in \mathbb{N}, m \geq l$ such that $|\alpha^{m!} \beta| \geq s$ and $|\gamma \delta^{m!}| \geq s$, it follows that

$$\widehat{\varphi}(\alpha^\omega \beta u \gamma \delta^\omega) = \varphi(\alpha^{m!} \beta u_k \gamma \delta^{m!}) = \varphi(\alpha^{m!} \beta v_k \gamma \delta^{m!}) = \widehat{\varphi}(\alpha^\omega \beta v \gamma \delta^\omega)$$

because $[u_k]_{\equiv_\varphi} = \widehat{\mu}(u) = \widehat{\mu}(v) = [v_k]_{\equiv_\varphi}$. Thus, we have that

$$\widehat{\varphi}(\alpha^\omega \beta u \gamma \delta^\omega) = \widehat{\varphi}(\alpha^\omega \beta v \gamma \delta^\omega)$$

for all $\alpha, \beta, \gamma, \delta \in \Sigma^+$.

Assume then that $\widehat{\varphi}(\alpha^\omega \beta u \gamma \delta^\omega) = \widehat{\varphi}(\alpha^\omega \beta v \gamma \delta^\omega)$ for all $\alpha, \beta, \gamma, \delta \in \Sigma^+$. Take any $\alpha, \beta \in \Sigma^{\geq s}$. Since $\varphi(\Sigma^s)$ is a finite semigroup and verifies that $\varphi(\Sigma^s) = \varphi(\Sigma^s)^2$, by a classical result in finite semigroup theory (see e.g. [14, Chapter 1, Proposition 1.12]), we have that there exist $\alpha_1, e, f, \beta_2 \in \Sigma^s$ and $\alpha_2, \beta_1 \in \Sigma^{\geq s}$ such that $\varphi(\alpha_1 e \alpha_2) = \varphi(\alpha)$ and $\varphi(\beta_1 f \beta_2) = \varphi(\beta)$ with $\varphi(e)$ and $\varphi(f)$ idempotents. Now, since $\varphi(e)$ is idempotent, we have that

$$\widehat{\varphi}(e^\omega) = \widehat{\varphi}(\lim_{n \rightarrow \infty} e^{n!}) = \lim_{n \rightarrow \infty} \varphi(e^{n!}) = \lim_{n \rightarrow \infty} \varphi(e)^{n!} = \varphi(e)$$

and similarly, $\widehat{\varphi}(f^\omega) = \varphi(f)$. So it follows that

$$\begin{aligned} \varphi(\alpha u_k \beta) &= \varphi(\alpha_1 e \alpha_2 u_k \beta_1 f \beta_2) \\ &= \widehat{\varphi}(\alpha_1 e^\omega \alpha_2 u \beta_1 f^\omega \beta_2) \\ &= \widehat{\varphi}(\alpha_1 e^\omega \alpha_2 v \beta_1 f^\omega \beta_2) \\ &= \varphi(\alpha_1 e \alpha_2 v_k \beta_1 f \beta_2) \\ &= \varphi(\alpha v_k \beta). \end{aligned}$$

As this is true for any $\alpha, \beta \in \Sigma^{\geq s}$, by definition it holds that $u_k \equiv_\varphi v_k$, hence $\widehat{\mu}(u) = \mu(u_k) = \mu(v_k) = \widehat{\mu}(v)$. \triangleleft

This concludes the proof of the proposition. \blacktriangleleft

4 Essentially- \mathbf{V} stamps and the join of \mathbf{V} and \mathbf{LI}

In this section, we establish the link between essentially- \mathbf{V} stamps and $\mathbf{V} \vee \mathbf{LI}$ and give a criterion that characterises exactly when they do correspond.

More precisely, consider the following criterion for a variety of monoids \mathbf{V} .

► **Criterion (A).** For any $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ with Σ an alphabet, we have $xLy \in \mathcal{L}(\mathbf{V} \vee \mathbf{LI})(\Sigma)$ for all $x, y \in \Sigma^*$.

It is a kind of mild closure condition that appears to be a sufficient and necessary condition for \mathbf{EV} and $\mathbf{V} \vee \mathbf{LI}$ to correspond.

► **Proposition 8.** *Let \mathbf{V} be a variety of monoids. Then $\langle \mathbf{V} \vee \mathbf{LI} \rangle_{\text{ne}} \subseteq \mathbf{EV}$ and equality holds if and only if \mathbf{V} verifies criterion (A).*

Why this proposition is useful to give characterisations of $\mathbf{V} \vee \mathbf{LI}$ in terms of identities will become clear in the next section. For now, we focus on its proof, that entirely relies on the following characterisation of the languages recognised by essentially- \mathbf{V} stamps.

► **Proposition 9.** *Let \mathbf{V} be a variety of monoids. For any alphabet Σ , the set $\mathcal{L}(\mathbf{EV})(\Sigma)$ consists of all Boolean combinations of languages of the form xLy for $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and $x, y \in \Sigma^*$.*

Proof. Let \mathcal{C} be the class of languages such that for any alphabet Σ , the set $\mathcal{C}(\Sigma)$ consists of all Boolean combinations of languages of the form xLy for $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and $x, y \in \Sigma^*$.

Let Σ be an alphabet. We need to show that $\mathcal{L}(\mathbf{EV})(\Sigma) = \mathcal{C}(\Sigma)$.

Inclusion from right to left. Let $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and $x, y \in \Sigma^*$. Let $\mu: \Sigma^* \rightarrow N$ be the syntactic morphism of L : this implies that $N \in \mathbf{V}$ and that there exists $F \subseteq N$ such that $L = \mu^{-1}(F)$. Let also $\varphi: \Sigma^* \rightarrow M$ be the syntactic morphism of the language $xLy = x\Sigma^*y \cap \Sigma^{|x|}\mu^{-1}(F)\Sigma^{|y|}$ and let s be its stability index. We then consider $u, v \in \Sigma^*$ such that $\mu(u) = \mu(v)$. Take any $x', y' \in \Sigma^*$ such that $|x'| \geq |x|$ and $|y'| \geq |y|$. We clearly have that $x'uy' \in x\Sigma^*y$ if and only if $x'vy' \in x\Sigma^*y$. Moreover, $x' = x'_1x'_2$ for $x'_1 \in \Sigma^{|x|}$ and $x'_2 \in \Sigma^*$ and $y' = y'_1y'_2$ for $y'_1 \in \Sigma^*$ and $y'_2 \in \Sigma^{|y|}$, so that

$$\begin{aligned} x'uy' \in \Sigma^{|x|}\mu^{-1}(F)\Sigma^{|y|} &\Leftrightarrow \mu(x'_2uy'_1) \in F \\ &\Leftrightarrow \mu(x'_2vy'_1) \in F \\ &\Leftrightarrow x'vy' \in \Sigma^{|x|}\mu^{-1}(F)\Sigma^{|y|} . \end{aligned}$$

Hence, $x'uy' \in xLy$ if and only if $x'vy' \in xLy$ for all $x', y' \in \Sigma^*$ such that $|x'| \geq |x|$ and $|y'| \geq |y|$, so that, by definition of the stability index s of φ and as φ is the syntactic morphism of xLy , we have $\varphi(x'uy') = \varphi(x'vy')$ for all $x', y' \in \Sigma^s$. Thus, it follows that $\varphi \in \mathbf{EV}$.

This implies that $xLy \in \mathcal{L}(\mathbf{EV})(\Sigma)$. Therefore, since this is true for any $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and $x, y \in \Sigma^*$ and since $\mathcal{L}(\mathbf{EV})(\Sigma)$ is closed under Boolean operations, we can conclude that $\mathcal{C}(\Sigma) \subseteq \mathcal{L}(\mathbf{EV})(\Sigma)$.

Inclusion from left to right. Let $L \in \mathcal{L}(\mathbf{EV})(\Sigma)$ and let $\varphi: \Sigma^* \rightarrow M$ be its syntactic morphism: it is an essentially- \mathbf{V} stamp. Given s its stability index, this means there exists a stamp $\mu: \Sigma^* \rightarrow N$ with $N \in \mathbf{V}$ such that for all $u, v \in \Sigma^*$, we have

$$\mu(u) = \mu(v) \Rightarrow (\varphi(xuy) = \varphi(xvy) \quad \forall x, y \in \Sigma^s) .$$

For each $m \in N$ and $x, y \in \Sigma^s$ consider the language $x\mu^{-1}(m)y$. For any two words $w, w' \in x\mu^{-1}(m)y$, we have $w = xuy$ and $w' = xvy$ with $\mu(u) = \mu(v) = m$, so that $\varphi(w) = \varphi(w')$. By definition of the syntactic morphism, this means that for all $m \in N$ and $x, y \in \Sigma^s$, either $x\mu^{-1}(m)y \subseteq L$ or $x\mu^{-1}(m)y \cap L = \emptyset$. Therefore, there exists a set $E \subseteq N \times \Sigma^s \times \Sigma^s$ such that $L \cap \Sigma^{\geq 2s} = \bigcup_{(m,x,y) \in E} x\mu^{-1}(m)y$, hence

$$L = \bigcup_{(m,x,y) \in E} x\mu^{-1}(m)y \cup F$$

for a certain $F \subseteq \Sigma^{<2s}$.

Take $w \in F$. We have that $\{w\} = w\Sigma^* \cap \bigcap_{a \in \Sigma} (\Sigma^* \setminus wa\Sigma^*)$ with $\Sigma^* \in \mathcal{L}(\mathbf{V})(\Sigma)$. Thus, the singleton language $\{w\}$ belongs to $\mathcal{C}(\Sigma)$ and since this is true for any $w \in F$ and F is finite, we can deduce from this that F is in $\mathcal{C}(\Sigma)$, as the latter is trivially closed under Boolean operations.

Now, for all $m \in N$, the language $\mu^{-1}(m)$ belongs to $\mathcal{L}(\mathbf{V})(\Sigma)$, so we finally have $L \in \mathcal{C}(\Sigma)$. This is true for any $L \in \mathcal{L}(\mathbf{EV})(\Sigma)$, so in conclusion, $\mathcal{L}(\mathbf{EV})(\Sigma) \subseteq \mathcal{C}(\Sigma)$. \blacktriangleleft

Proposition 8 then follows from the two next lemmata, that are both easy consequences of Proposition 9. For completeness, we give the proofs in the appendix.

► **Lemma 10.** *Let \mathbf{V} be a variety of monoids. Then $\langle \mathbf{V} \vee \mathbf{LI} \rangle_{\text{ne}} \subseteq \mathbf{EV}$.*

► **Lemma 11.** *Let \mathbf{V} be a variety of monoids. Then $\mathbf{EV} \subseteq \langle \mathbf{V} \vee \mathbf{LI} \rangle_{\text{ne}}$ if and only if \mathbf{V} verifies criterion (A).*

5 Applications

In this last section, we use the link between essentially- \mathbf{V} stamps and $\mathbf{V} \vee \mathbf{LI}$ to reprove some characterisations of joins between \mathbf{LI} and some well-known varieties of monoids in terms of identities.

One thing seems at first glance a bit problematic about proving that a variety of monoids \mathbf{V} satisfies criterion (A). Indeed, to this end, one needs to prove that certain languages belong to $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})$; however, this poses a problem when one's goal is precisely to characterise $\mathbf{V} \vee \mathbf{LI}$, because one shall a priori not know more about $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})$ than what is given by Proposition 2. Nevertheless, there is a natural sufficient condition for criterion (A) to hold that depends only on $\mathcal{L}(\mathbf{V})$: if given any language $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and any $x, y \in \Sigma^*$ with Σ an alphabet, there exists a language $K \in \mathcal{L}(\mathbf{V})(\Sigma)$ such that L is equal to the quotient $x^{-1}Ky^{-1}$, then \mathbf{V} verifies criterion (A). We don't know whether this quotient-expressibility condition that solely depends on the variety \mathbf{V} (without explicit reference to \mathbf{LI}) is actually equivalent to it satisfying criterion (A), but we can prove such an equivalence for a weaker quotient-expressibility condition for \mathbf{V} . The proof is to be found in the appendix.

► **Proposition 12.** *Let \mathbf{V} be a variety of monoids. Then \mathbf{V} satisfies criterion (A) if and only if for any $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and any $x, y \in \Sigma^*$ with Σ an alphabet, there exist $k, l \in \mathbb{N}$ such that for all $u \in \Sigma^k, v \in \Sigma^l$, there exists a language $K \in \mathcal{L}(\mathbf{V})(\Sigma)$ verifying $u^{-1}Lv^{-1} = (xu)^{-1}K(vy)^{-1}$.*

This quotient-expressibility condition appears to be particularly useful to prove that a variety of monoids \mathbf{V} does not satisfy criterion (A) without needing to understand what $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})$ is. We demonstrate this for the variety of finite commutative and idempotent monoids \mathbf{J}_1 .

► **Proposition 13.** \mathbf{J}_1 does not satisfy criterion (A).

Proof. Given an alphabet Σ , the set $\mathcal{L}(\mathbf{J}_1)(\Sigma)$ consists of all Boolean combinations of languages of the form $\Sigma^*a\Sigma^*$ for $a \in \Sigma$ (see [14, Chapter 2, Proposition 3.10]).

Let $L = \{a, b\}^*b\{a, b\}^* \in \mathcal{L}(\mathbf{J}_1)(\{a, b\})$ and $x = b, y = \varepsilon$. Take any $k, l \in \mathbb{N}$ and set $u = a^k$ and $v = a^l$. Consider a $K \in \mathcal{L}(\mathbf{J}_1)(\{a, b\})$. We have that $xuavy \in K \Leftrightarrow xuabvy \in K$ so that $a \in (xu)^{-1}K(vy)^{-1} \Leftrightarrow ab \in (xu)^{-1}K(vy)^{-1}$. But $a \notin u^{-1}Lv^{-1}$ and $ab \in u^{-1}Lv^{-1}$, hence $u^{-1}Lv^{-1} \neq (xu)^{-1}K(vy)^{-1}$ and this holds for any choice of K . So for any $k, l \in \mathbb{N}$, there exists $u \in \Sigma^k, v \in \Sigma^l$ such that no $K \in \mathcal{L}(\mathbf{J}_1)(\{a, b\})$ verifies $u^{-1}Lv^{-1} = (xu)^{-1}K(vy)^{-1}$.

In conclusion, by Proposition 12, \mathbf{J}_1 does not satisfy criterion (A). ◀

We now prove the announced characterisations of joins between \mathbf{LI} and some well-known varieties of monoids in terms of identities.

► **Theorem 14.** *We have the following.*

1. $\langle \mathbf{R} \vee \mathbf{LI} \rangle_{\text{ne}} = \mathbf{ER} = \llbracket x^\omega y(ab)^\omega azt^\omega = x^\omega y(ab)^\omega zt^\omega \rrbracket_{\text{ne}}$.
2. $\langle \mathbf{L} \vee \mathbf{LI} \rangle_{\text{ne}} = \mathbf{EL} = \llbracket x^\omega yb(ab)^\omega zt^\omega = x^\omega y(ab)^\omega zt^\omega \rrbracket_{\text{ne}}$.
3. $\langle \mathbf{J} \vee \mathbf{LI} \rangle_{\text{ne}} = \mathbf{EJ} = \llbracket x^\omega y(ab)^\omega azt^\omega = x^\omega y(ab)^\omega zt^\omega, x^\omega yb(ab)^\omega zt^\omega = x^\omega y(ab)^\omega zt^\omega \rrbracket_{\text{ne}}$.
4. $\langle \mathbf{H} \vee \mathbf{LI} \rangle_{\text{ne}} = \mathbf{EH}$ for any variety of groups \mathbf{H} .

Proof. In each case, we prove that the variety of monoids under consideration satisfies criterion (A) using Proposition 12. We then use Propositions 8 and 6.

Proof of 1. It is well-known that given an alphabet Σ , the set $\mathcal{L}(\mathbf{R})(\Sigma)$ consists of all languages that are disjoint unions of languages that are of the form $A_0^*a_1A_1^*\cdots a_kA_k^*$ where $k \in \mathbb{N}$, $a_1, \dots, a_k \in \Sigma$, $A_0, A_1, \dots, A_k \subseteq \Sigma$ and $a_i \notin A_{i-1}$ for all $i \in [k]$ (see [14, Chapter 4, Theorem 3.3]).

Let Σ be an alphabet and take a language $A_0^*a_1A_1^*\cdots a_kA_k^*$ where $k \in \mathbb{N}$, $a_1, \dots, a_k \in \Sigma$, $A_0, A_1, \dots, A_k \subseteq \Sigma$ and $a_i \notin A_{i-1}$ for all $i \in [k]$. Take $x, y \in \Sigma^*$. Observe that y can be uniquely written as $y = zt$ where $z \in A_k^*$ and $t \in \{\varepsilon\} \cup (\Sigma \setminus A_k)\Sigma^*$. We have

$$A_0^*a_1A_1^*\cdots a_kA_k^* = x^{-1} \left(xA_0^*a_1A_1^*\cdots a_kA_k^*t \cap \bigcap_{v \in A_k^{<|z|}} (\Sigma^* \setminus xA_0^*a_1A_1^*\cdots a_kv t) \right) y^{-1}$$

using the convention that $xA_0^*a_1A_1^*\cdots a_kv t = xvt$ for all $v \in A_k^{<|z|}$ when $k = 0$. The language $xA_0^*a_1A_1^*\cdots a_kA_k^*t \cap \bigcap_{v \in A_k^{<|z|}} (\Sigma^* \setminus xA_0^*a_1A_1^*\cdots a_kv t)$ does belong to the set $\mathcal{L}(\mathbf{R})(\Sigma)$ because the latter is closed under Boolean operations and by definition of z and t . Thus, we can conclude that for each $L \in \mathcal{L}(\mathbf{R})(\Sigma)$ and $x, y \in \Sigma^*$, there exists $K \in \mathcal{L}(\mathbf{R})(\Sigma)$ such that $L = x^{-1}Ky^{-1}$ by using the characterisation of $\mathcal{L}(\mathbf{R})(\Sigma)$, the fact that quotients commute with unions [14, p. 20] and closure of $\mathcal{L}(\mathbf{R})(\Sigma)$ under unions.

Proof of 2. It is also well-known that given an alphabet Σ , the set $\mathcal{L}(\mathbf{L})(\Sigma)$ consists of all languages that are disjoint unions of languages that are of the form $A_0^*a_1A_1^*\cdots a_kA_k^*$ where $k \in \mathbb{N}$, $a_1, \dots, a_k \in \Sigma$, $A_0, A_1, \dots, A_k \subseteq \Sigma$ and $a_i \notin A_i$ for all $i \in [k]$ (see [14, Chapter 4, Theorem 3.4]). The proof is then dual to the previous case.

Proof of 3. Given an alphabet Σ , for each $k \in \mathbb{N}$, we define the equivalence relation \sim_k on Σ^* by $u \sim_k v$ for $u, v \in \Sigma^*$ whenever u and v have the same set of subwords of length at most k . This relation is a congruence of finite index on Σ^* . Simon proved [19] that a language belongs to $\mathcal{L}(\mathbf{J})(\Sigma)$ if and only if it is equal to a union of \sim_k -classes for a $k \in \mathbb{N}$.

Let Σ be an alphabet and take $L \in \mathcal{L}(\mathbf{J})(\Sigma)$ as well as $x, y \in \Sigma^*$. Thus, there exists $k \in \mathbb{N}$ such that L is a union of \sim_k -classes. Define the language $K = \bigcup_{w \in L} [xwy]_{\sim_{|xy|+k}}$: it belongs to $\mathcal{L}(\mathbf{J})(\Sigma)$ by construction. We now show that $L = x^{-1}Ky^{-1}$, which concludes the proof. Let $w \in L$: we have that $xwy \in [xwy]_{\sim_{|xy|+k}} \subseteq K$, so that $w \in x^{-1}Ky^{-1}$. Let conversely $w \in x^{-1}Ky^{-1}$. This means that $xwy \in K$, which implies that there exists $w' \in L$ such that $xwy \sim_{|xy|+k} xw'y$. Actually, it holds that any $u \in \Sigma^*$ of length at most k is a subword of w if and only if it is a subword of w' , because xuy is a subword of xwy if and only if it is a subword of $xw'y$. Hence, $w \sim_k w'$, which implies that $w \in L$.

Proof of 4. Consider any variety of groups \mathbf{H} . Take a language $L \in \mathcal{L}(\mathbf{H})(\Sigma)$ for an alphabet Σ and let $x, y \in \Sigma^*$. Consider the syntactic morphism $\eta: \Sigma^* \rightarrow M$ of L : we have that M is a group in \mathbf{H} . Define the language $K = \eta^{-1}(\eta(x)\eta(L)\eta(y))$: it belongs to $\mathcal{L}(\mathbf{H})(\Sigma)$. We now show that $L = x^{-1}Ky^{-1}$, which concludes the proof. Let $w \in L$: we have that $\eta(xwy) \in \eta(x)\eta(L)\eta(y)$, so that $w \in x^{-1}Ky^{-1}$. Conversely, let $w \in x^{-1}Ky^{-1}$. We have that $xwy \in K$, which means that $\eta(xwy) = \eta(x)\eta(w')\eta(y)$ for a $w' \in L$, so that $\eta(w) = \eta(w') \in \eta(L)$, as any element in M is invertible. Thus, $w \in L$. ◀

6 Conclusion

The general method presented in this paper actually allows to reprove in a straightforward language-theoretic way even more characterisations of the join of **LI** with some variety of finite monoids. This can for instance be done for the variety of finite commutative monoids **Com** or the variety of finite commutative aperiodic monoids **ACom**.

In fact, as already observed in some sense by Costa [9], many varieties of finite monoids seem to verify criterion (A). The main question left open by this present work is to understand better what exactly those varieties are. Another question left open is whether Proposition 12 can be refined by using the stronger quotient-expressibility condition alluded to before the statement of the proposition. The answers to both questions are unclear to the author, but making progress on them may also lead to a better understanding of joins of varieties of finite monoids with **LI**.

References

- 1 Douglas Albert, Robert Baldinger, and John Rhodes. Undecidability of the identity problem for finite semigroups. *J. Symb. Log.*, 57(1):179–192, 1992. doi:10.2307/2275184.
- 2 Jorge Almeida. Some pseudovariety joins involving the pseudovariety of finite groups. *Semigroup Forum*, 37(1):53–57, 1988. doi:10.1007/bf02573123.
- 3 Jorge Almeida. *Finite Semigroups and Universal Algebra*, volume 3. WORLD SCIENTIFIC, 1995. doi:10.1142/2481.
- 4 Jorge Almeida and Assis Azevedo. The join of the pseudovarieties of R-trivial and L-trivial monoids. *Journal of Pure and Applied Algebra*, 60(2):129–137, 1989. doi:10.1016/0022-4049(89)90125-4.
- 5 Jorge Almeida and Pascal Weil. Free profinite \mathcal{R} -trivial monoids. *Int. J. Algebra Comput.*, 7(5):625–672, 1997. doi:10.1142/S0218196797000289.
- 6 Assis Azevedo. The join of the pseudovariety J with permutative pseudovarieties. In *Lattices, Semigroups, and Universal Algebra*, pages 1–11. Springer US, 1990. doi:10.1007/978-1-4899-2608-1_1.
- 7 Assis Azevedo and Marc Zeitoun. Three examples of join computations. *Semigroup Forum*, 57(2):249–277, 1998. doi:10.1007/p100005976.
- 8 Laura Chaubard, Jean-Éric Pin, and Howard Straubing. First order formulas with modular predicates. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 211–220. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.24.
- 9 José Carlos Costa. Some pseudovariety joins involving locally trivial semigroups. *Semigroup Forum*, 64(1):12–28, 2001. doi:10.1007/s002330010060.
- 10 José Carlos Costa. Some pseudovariety joins involving groups and locally trivial semigroups. In *Semigroups, Algorithms, Automata and Languages*, pages 341–348. WORLD SCIENTIFIC, 2002. doi:10.1142/9789812776884_0013.
- 11 Samuel Eilenberg. *Automata, Languages, and Machines. A*. Pure and applied mathematics. Academic Press, New York, 1974. URL: <https://www.worldcat.org/oclc/310535248>.
- 12 Samuel Eilenberg. *Automata, Languages, and Machines. B*. Pure and applied mathematics. Academic Press, New York, 1976. URL: <https://www.worldcat.org/oclc/310535259>.
- 13 Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. Tameness and the power of programs over monoids in DA. *CoRR*, abs/2101.07495, 2021. arXiv:2101.07495.
- 14 Jean-Éric Pin. *Varieties of Formal Languages*. North Oxford, London and Plenum, New-York, 1986. (Traduction de Variétés de langages formels).
- 15 Jean-Éric Pin. Syntactic semigroups. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*, pages 679–746. Springer, 1997. doi:10.1007/978-3-642-59136-5_10.
- 16 Jean-Éric Pin and Howard Straubing. Some results on \mathcal{C} -varieties. *RAIRO Theor. Informatics Appl.*, 39(1):239–262, 2005. doi:10.1051/ita:2005014.
- 17 Jan Reiterman. The Birkhoff theorem for finite algebras. *Algebra Universalis*, 14(1):1–10, 1982. doi:10.1007/bf02483902.
- 18 Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Inf. Control.*, 8(2):190–194, 1965. doi:10.1016/S0019-9958(65)90108-7.

- 19 Imre Simon. Piecewise testable events. In H. Barkhage, editor, *Automata Theory and Formal Languages, 2nd GI Conference, Kaiserslautern, May 20-23, 1975*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer, 1975. doi:10.1007/3-540-07407-4_23.
- 20 Howard Straubing. On logical descriptions of regular languages. In Sergio Rajsbaum, editor, *LATIN 2002: Theoretical Informatics, 5th Latin American Symposium, Cancun, Mexico, April 3-6, 2002, Proceedings*, volume 2286 of *Lecture Notes in Computer Science*, pages 528–538. Springer, 2002. doi:10.1007/3-540-45995-2_46.
- 21 Marc Zeitoun. The join of the pseudovarieties of idempotent semigroups and locally trivial semigroups. *Semigroup Forum*, 50(1):367–381, 1995. doi:10.1007/bf02573532.
- 22 Marc Zeitoun. On the decidability of the membership problem of the pseudovariety JvB. *Int. J. Algebra Comput.*, 5(1):47–64, 1995. doi:10.1142/S0218196795000057.
- 23 Marc Zeitoun. On the join of two pseudovarieties. *Semigroups, Automata and Languages*, eds. J. Almeida, GMS Gomes, and PV Silva. *World Scientific*, pages 281–288, 1996.

A Missing proofs

Proof of Proposition 2. Let \mathbf{W} be an ne -variety of stamps such that $\langle \mathbf{V} \rangle_{all} \cup \langle \mathbf{LI} \rangle_{ne} \subseteq \mathbf{W}$. There exists a variety of semigroups \mathbf{W}' such that $\langle \mathbf{W}' \rangle_{ne} = \mathbf{W}$.

Let $S \in \mathbf{V} \cup \mathbf{LI}$. We denote by S^1 the monoid S if S is already a monoid and the monoid $S \cup \{1\}$ otherwise. Then the evaluation morphism $\eta_S: S^* \rightarrow S^1$ such that $\eta_S(s) = s$ for all $s \in S$ verifies $\eta_S(S^+) = S$ and additionally $S^1 = S$ when $S \in \mathbf{V}$. This implies that $\eta_S \in \langle \mathbf{V} \rangle_{all} \cup \langle \mathbf{LI} \rangle_{ne} \subseteq \mathbf{W}$. But by definition of \mathbf{W}' , it must be that $S = \eta_S(S^+) \in \mathbf{W}'$.

Therefore, \mathbf{W}' contains both \mathbf{V} and \mathbf{LI} , which implies that $\mathbf{V} \vee \mathbf{LI} \subseteq \mathbf{W}'$ by inclusion-wise minimality of $\mathbf{V} \vee \mathbf{LI}$. By definition, we can then conclude that $\langle \mathbf{V} \vee \mathbf{LI} \rangle_{ne} \subseteq \langle \mathbf{W}' \rangle_{ne} = \mathbf{W}$. So $\langle \mathbf{V} \vee \mathbf{LI} \rangle_{ne}$ is the inclusion-wise least ne -variety of stamps containing both $\langle \mathbf{V} \rangle_{all}$ and $\langle \mathbf{LI} \rangle_{ne}$.

Let now \mathcal{W} be an ne -variety of languages such that $\mathcal{L}(\mathbf{V}) \cup \mathcal{L}(\mathbf{LI}) \subseteq \mathcal{W}$. It holds that $\mathcal{W} = \mathcal{L}(\mathbf{W})$ for an ne -variety of stamps \mathbf{W} . We have that $\langle \mathbf{V} \rangle_{all}$, which is in particular an ne -variety of stamps, is included in \mathbf{W} because $\mathcal{L}(\langle \mathbf{V} \rangle_{all}) = \mathcal{L}(\mathbf{V}) \subseteq \mathcal{W} = \mathcal{L}(\mathbf{W})$, but also that $\langle \mathbf{LI} \rangle_{ne}$ is included in \mathbf{W} because $\mathcal{L}(\langle \mathbf{LI} \rangle_{ne}) = \mathcal{L}(\mathbf{LI}) \subseteq \mathcal{W} = \mathcal{L}(\mathbf{W})$. By inclusion-wise minimality of $\langle \mathbf{V} \vee \mathbf{LI} \rangle_{ne}$, it follows that $\langle \mathbf{V} \vee \mathbf{LI} \rangle_{ne} \subseteq \mathbf{W}$. Hence, using again the above fact on the Eilenberg correspondence, we can conclude that $\mathcal{L}(\mathbf{V} \vee \mathbf{LI}) = \mathcal{L}(\langle \mathbf{V} \vee \mathbf{LI} \rangle_{ne}) \subseteq \mathcal{L}(\mathbf{W}) = \mathcal{W}$. So $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})$ is the inclusion-wise least ne -variety of languages containing both $\mathcal{L}(\mathbf{V})$ and $\mathcal{L}(\mathbf{LI})$.

Consider now the class of languages \mathcal{C} such that $\mathcal{C}(\Sigma)$ is the Boolean closure of $\mathcal{L}(\mathbf{V})(\Sigma) \cup \mathcal{L}(\mathbf{LI})(\Sigma)$ for each alphabet Σ . By closure under Boolean operations of $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})$, we have that $\mathcal{C} \subseteq \mathcal{L}(\mathbf{V} \vee \mathbf{LI})$. Now, as Boolean operations commute with both quotients [14, p. 20] and inverses of ne -morphisms [14, Proposition 0.4], by closure of $\mathcal{L}(\mathbf{V})$ and $\mathcal{L}(\mathbf{LI})$ under quotients and inverses of ne -morphisms, we actually have that \mathcal{C} is an ne -variety of languages. Therefore, by inclusion-wise minimality of $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})$, we can conclude that $\mathcal{L}(\mathbf{V} \vee \mathbf{LI}) = \mathcal{C}$. ◀

Proof of Lemma 10. We actually have that $\mathcal{L}(\mathbf{V}) \cup \mathcal{L}(\mathbf{LI}) \subseteq \mathcal{L}(\mathbf{EV})$, which allows us to conclude by inclusion-wise minimality of $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})$ (Proposition 2) and by the fact that $\mathcal{L}(\mathbf{EV})$ is an ne -variety of languages (Proposition 6).

Let Σ be an alphabet. The fact that $\mathcal{L}(\mathbf{V})(\Sigma) \subseteq \mathcal{L}(\mathbf{EV})(\Sigma)$ follows trivially from Proposition 9. Moreover, for all $u \in \Sigma^*$, since necessarily $\Sigma^* \in \mathcal{L}(\mathbf{V})(\Sigma)$, we have that both $u\Sigma^*$ and Σ^*u belong to $\mathcal{L}(\mathbf{LI})(\Sigma)$. Thus, as $\mathcal{L}(\mathbf{EV})(\Sigma)$ is closed under Boolean operations, it follows that $\mathcal{L}(\mathbf{LI})(\Sigma) \subseteq \mathcal{L}(\mathbf{EV})(\Sigma)$.

This concludes the proof, since it holds for any alphabet Σ . ◀

Proof of Lemma 11. Assume that $\mathbf{EV} \subseteq \langle \mathbf{V} \vee \mathbf{LI} \rangle_{ne}$. For any $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and any $x, y \in \Sigma^*$ with Σ an alphabet, by Proposition 9, we have that $xLy \in \mathcal{L}(\mathbf{EV})(\Sigma) \subseteq \mathcal{L}(\mathbf{V} \vee \mathbf{LI})(\Sigma)$. Hence, \mathbf{V} verifies criterion (A).

Conversely, assume that \mathbf{V} verifies criterion (A). For any alphabet Σ , the set $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})(\Sigma)$ contains all languages of the form xLy for $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and $x, y \in \Sigma^*$, so it contains all Boolean combinations of languages of that form, since it is closed under Boolean operations. Therefore, by Proposition 9, we have $\mathcal{L}(\mathbf{EV}) \subseteq \mathcal{L}(\mathbf{V} \vee \mathbf{LI})$, so that $\mathbf{EV} \subseteq \langle \mathbf{V} \vee \mathbf{LI} \rangle_{ne}$. ◀

Proof of Proposition 12. Let us first observe that given any alphabet Σ , given any language K on that alphabet and given any two words $x, y \in \Sigma^*$, we have that $x(x^{-1}Ky^{-1})y = x\Sigma^*y \cap K$ and $x^{-1}(xKy)y^{-1} = K$.

Implication from right to left. Assume that for any $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ and any $x, y \in \Sigma^*$ with Σ an alphabet, there exist $k, l \in \mathbb{N}$ such that for all $u \in \Sigma^k, v \in \Sigma^l$, there exists a language $K \in \mathcal{L}(\mathbf{V})(\Sigma)$ verifying $u^{-1}Lv^{-1} = (xu)^{-1}K(vy)^{-1}$. Take $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ for an alphabet Σ and take $x, y \in \Sigma^*$. Consider also $k, l \in \mathbb{N}$ that are guaranteed to exist by the assumption we just made.

For all $u \in \Sigma^k, v \in \Sigma^l$, there exists a language $K \in \mathcal{L}(\mathbf{V})(\Sigma)$ verifying $u^{-1}Lv^{-1} = (xu)^{-1}K(vy)^{-1}$, so that by our observation at the beginning of the proof, we have

$$x(u\Sigma^*v \cap L)y = xu(u^{-1}Lv^{-1})vy = xu((xu)^{-1}K(vy)^{-1})vy = xu\Sigma^*vy \cap K .$$

Using Proposition 2, we thus have that $x(u\Sigma^*v \cap L)y \in \mathcal{L}(\mathbf{V} \vee \mathbf{LI})(\Sigma)$ for all $u \in \Sigma^k, v \in \Sigma^l$. Moreover, since we have that the set of words of L of length at least $k + l$ is

$$\Sigma^{\geq k+l} \cap L = \bigcup_{u \in \Sigma^k, v \in \Sigma^l} (u\Sigma^*v \cap L)$$

and since

$$L = (\Sigma^{\geq k+l} \cap L) \cup F$$

where F is a finite set of words on Σ of length less than $k + l$, we have that

$$xLy = x((\Sigma^{\geq k+l} \cap L) \cup F)y = \bigcup_{u \in \Sigma^k, v \in \Sigma^l} x(u\Sigma^*v \cap L)y \cup xFy .$$

We can thus conclude that $xLy \in \mathcal{L}(\mathbf{V} \vee \mathbf{LI})(\Sigma)$ since $xFy \in \mathcal{L}(\mathbf{LI})(\Sigma)$ and because $\mathcal{L}(\mathbf{V} \vee \mathbf{LI})(\Sigma)$ is closed under unions.

Implication from left to right. Assume that \mathbf{V} satisfies criterion (A). Take $L \in \mathcal{L}(\mathbf{V})(\Sigma)$ for an alphabet Σ and take $x, y \in \Sigma^*$. By hypothesis, we know that $xLy \in \mathcal{L}(\mathbf{V} \vee \mathbf{LI})(\Sigma)$.

By Proposition 2, this means that xLy is a Boolean combination of languages in $\mathcal{L}(\mathbf{V})(\Sigma) \cup \mathcal{L}(\mathbf{LI})(\Sigma)$. Further, this implies that xLy can be written as the union of intersections of languages of $\mathcal{L}(\mathbf{V})(\Sigma)$ and $\mathcal{L}(\mathbf{LI})(\Sigma)$ or their complements, which in turn implies, by closure of $\mathcal{L}(\mathbf{V})(\Sigma)$ and $\mathcal{L}(\mathbf{LI})(\Sigma)$ under Boolean operations, that xLy can be written as a finite union of languages of the form $K \cap (U\Sigma^*V \cup W)$ with $K \in \mathcal{L}(\mathbf{V})(\Sigma)$ and $U, V, W \subseteq \Sigma^*$ finite. Since any word in xLy must be of length at least $|xy|$ and have x as a prefix and y as a suffix, we can assume that any language $K \cap (U\Sigma^*V \cup W)$ appearing in a finite union as described above verifies that $U \subseteq x\Sigma^*$, that $V \subseteq \Sigma^*y$ and that $W \subseteq x\Sigma^*y$. Now, if we take $k, l \in \mathbb{N}$ big enough, we thus have that

51:16 A Note on the Join of Varieties of Monoids with LI

$$xLy = \bigcup_{u \in \Sigma^k, v \in \Sigma^l} (K_{u,v} \cap xu\Sigma^*vy) \cup F$$

where $K_{u,v} \in \mathcal{L}(\mathbf{V})(\Sigma)$ for all $u \in \Sigma^k, v \in \Sigma^l$ and $F \subseteq \Sigma^{<|xy|+k+l}$. Hence, for all $u \in \Sigma^k, v \in \Sigma^l$, we have

$$\begin{aligned} u^{-1}Lv^{-1} &= u^{-1}(x^{-1}(xLy)y^{-1})v^{-1} \\ &= (xu)^{-1} \left(\bigcup_{u' \in \Sigma^k, v' \in \Sigma^l} (K_{u',v'} \cap xu'\Sigma^*v'y) \cup F \right) (vy)^{-1} \\ &= \bigcup_{u' \in \Sigma^k, v' \in \Sigma^l} (xu)^{-1} \left(xu'((xu')^{-1}K_{u',v'}(v'y)^{-1})v'y \right) (vy)^{-1} \cup \\ &\quad (xu)^{-1}F(vy)^{-1} \\ &= (xu)^{-1}K_{u,v}(vy)^{-1}, \end{aligned}$$

using classical formulae for quotients [14, p. 20] and observing that $(xu)^{-1}K(vy)^{-1} = \emptyset$ for any $K \subseteq \Sigma^*$ such that $K \cap xu\Sigma^*vy = \emptyset$. ◀

Optimal Regular Expressions for Palindromes of Given Length

Hermann Gruber 

Knowledgepark GmbH, München, Germany

Markus Holzer¹ 

Institut für Informatik, University of Giessen, Germany

Abstract

The language P_n (\tilde{P}_n , respectively) consists of all words that are palindromes of length $2n$ ($2n - 1$, respectively) over a fixed binary alphabet. We construct a regular expression that specifies P_n (\tilde{P}_n , respectively) of alphabetic width $4 \cdot 2^n - 4$ ($3 \cdot 2^n - 4$, respectively) and show that this is optimal, that is, the expression has minimum alphabetic width among all expressions that describe P_n (\tilde{P}_n , respectively). To this end we give optimal expressions for the first k palindromes in lexicographic order of odd and even length, proving that the optimal bound is $2n + 4(k - 1) - 2S_2(k - 1)$ in case of odd length and $2n + 3(k - 1) - 2S_2(k - 1) - 1$ for even length, respectively. Here $S_2(n)$ refers to the Hamming weight function, which denotes the number of ones in the binary expansion of the number n .

2012 ACM Subject Classification Theory of computation → Regular languages; Mathematics of computing → Nonlinear equations

Keywords and phrases regular expression, descriptive complexity, lower bound, upper bound, recurrence, sum of digits

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.52

Acknowledgements We would like to thank the anonymous reviewers for their valuable suggestions.

1 Introduction

During the last two decades or so, literally hundreds of research papers have been investigating deterministic and nondeterministic state complexity of regular languages. Here, general purpose lower bound techniques are available, and in many cases, upper and lower bounds can be obtained that match exactly, not only asymptotically. For recent surveys, see, e.g., [8, 15].

The situation is less desirable if we investigate the minimum required size of regular expressions describing a regular language. While several different lower bound techniques are available, often the best known upper and lower bounds match only asymptotically. For illustration, the size blow-up when going from finite automata over a binary alphabet to regular expressions is at least c^n for some $c > 1$ for large enough n , cf. [12]. The current record holder for the upper bound is $O(1.682^n)$, see [5]. This gives a “tight” bound of $2^{\Theta(n)}$, which is on closer inspection a bit loose. To our knowledge, exactly matching upper and lower bounds for the minimum required expression size are known only for very few nontrivial language families: Namely, the Boolean n -bit parity function [7, 14], the less-than relation on an n -set [2], given as $\{ij \mid 1 \leq i < j \leq n\}$, and the permutations of an n -set [23].

The set of all palindromes over the alphabet $\{a, b\}$ is context-free but not regular; virtually every computer science student in the world will learn this during their curriculum. Not surprisingly, this basic observation is as old as the Chomsky hierarchy itself [3]. Of course, if we consider only palindromes of a given length, the set thus obtained is finite, and therefore

¹ Corresponding author.



regular. We exactly determine the optimum regular expressions for this set, for every given length. In the course of the proof, we also determine the optimum regular expressions for the lexicographically first k palindromes of a given length, for every k . The difficulty of course lies in establishing a matching lower bound. To this end, we use and expand a method from [23] to obtain a recurrent lower bound. The recurrence thus obtained involves a “minvolution” in the sense of [11] and the minimum operator of course yields a nonlinear recurrence. A long line of research concerns asymptotic and exact solutions of recurrences involving minimum and maximum functions, see, e.g., [18] and references therein. Our recurrence falls into neither of the known categories. So we develop a tailor-made strategy for solving the recurrence, and derive a novel identity involving sums of Hamming weights. We hope that this will serve as a helpful example for researchers in need of solving similar nonlinear recurrences.

Some of our results contribute to the knowledge about integer sequences: We give a characterization of the number of multiplications to compute the $(n + 1)$ th power by the ancient Indian Chandah-sutra method in terms of Hamming weights (Lemma 8). Also, we find a new recurrence for the numbers having a partition into distinct Mersenne numbers greater than zero (Lemma 10). The functions giving the optimal lengths of the regular expressions we consider can be enumerated in lexicographic order; accompanying submissions to the On-line Encyclopedia of Integer Sequences (OEIS) are in preparation, since these sequences are not yet covered by OEIS.

With some extra effort, all of our results can be generalized to larger alphabet sizes. These results will be presented in the full version of this paper.

2 Preliminaries

We assume that the reader is familiar with the basic notions of formal language theory as contained in [16]. In particular, let Σ be an *alphabet* and Σ^* the *set of all words over the alphabet* Σ , including the *empty word* ϵ . The *length of a word* w is denoted by $|w|$, where $|\epsilon| = 0$, and the total number of occurrences of the alphabet symbol a in w is denoted by $|w|_a$. In this paper, we mainly deal with finite languages. The *order* of a finite language L is the length of a longest word belonging to L . A finite language L is *homogeneous* if all words in the language have the same length. In order to fix the notation, we briefly recall the definition of regular expressions and the languages described by them.

The *regular expressions* over an alphabet Σ are defined inductively in the usual way:² \emptyset , ϵ , and every letter a with $a \in \Sigma$ is a regular expression; and when E and F are regular expressions, then $(E + F)$, $(E \cdot F)$, and $(E)^*$ are also regular expressions. The language defined by a regular expression E , denoted by $L(E)$, is defined as follows: $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$, $L(a) = \{a\}$, $L(E + F) = L(E) \cup L(F)$, $L(E \cdot F) = L(E) \cdot L(F)$, and $L(E^*) = L(E)^*$. The *alphabetic width* or *size* of a regular expression E over the alphabet Σ , denoted by $\text{awidth}(E)$, is defined as the total number of occurrences of letters of Σ in E . For a regular language L , we define its alphabetic width, $\text{awidth}(L)$, as the minimum alphabetic width among all regular expressions describing L .

² For convenience, parentheses in regular expressions are sometimes omitted and the concatenation is simply written as juxtaposition. The priority of operators is specified in the usual fashion: concatenation is performed before union, and star before both product and union.

3 A Lower Bound for Palindromes of Even Length

For a nonnegative integer n , let $P_n = \{ww^R \mid w \in \{a, b\}^n\}$ denote the set of palindromes of length $2n$. In this section, we give a tight bound on the required regular expression size of P_n . For our toolbox, we need to investigate the concatenation of homogeneous languages.

► **Lemma 1.** *Let L_1 and L_2 be homogeneous languages. Then $\text{awidth}(L_1 \cdot L_2) = \text{awidth}(L_1) + \text{awidth}(L_2)$.*

Inspired by the method recently used to exactly determine the alphabetic width of the set of permutations [23], define $\ell(n, k)$ to be the minimum alphabetic width of a regular expression describing a subset of P_n , where the subset has cardinality at least k . Note that $\ell(n, k)$ is monotone in k by definition, that is, $\ell(n, k) \leq \ell(n, k')$, for $k \leq k'$.

► **Lemma 2.** *Let $n \geq 0$ and $1 \leq k \leq 2^n$. Then $\ell(n, k)$ obeys the following recurrence:*

$$\ell(n, k) \geq \min\{\ell(n-1, k) + 2, \min_{1 \leq i < k} \{\ell(n, i) + \ell(n, k-i)\}\}, \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

$$\ell(n, k) \geq \min_{1 \leq i < k} \{\ell(n, i) + \ell(n, k-i)\}, \text{ for } n \geq 1 \text{ and } k > 2^{n-1},$$

and

$$\ell(n, 1) = 2n.$$

Proof. In the case $k = 1$, each regular expression describing a nonempty subset of $\{a, b\}^{2n}$ must have alphabetic width at least $2n$. For $n \geq 1$, the expression a^{2n} describes at least one word in P_n . For $n = 0$, ϵ is an optimal regular expression describing the only nonempty subset of $\{a, b\}^0 = \{\epsilon\}$. Thus we have $\ell(n, 1) = 2n$ for all $n \geq 0$.

For $n \geq 1$ and $2 \leq k \leq 2^n$, let E be a regular expression denoting a subset of P_n which has cardinality at least k . The language P_n is homogeneous, so we may safely assume that neither ϵ nor \emptyset occur in E , and the same holds for the Kleene star, see, e.g., [14]. Thus, E is of the form $F + G$ or of the form $F \cdot G$, and each of F and G have alphabetic width at least 1.

If $E = F + G$, then both F and G denote subsets of P_n , say of sizes k_1 and k_2 , respectively. Then $k_1 + k_2 \geq k$, and, by minimality, $k_1, k_2 < k$. We thus obtain the following recurrence in the case of union:

$$\begin{aligned} \ell(n, k) &\geq \ell(n, k_1) + \ell(n, k_2) \\ &\geq \ell(n, k_1) + \ell(n, k - k_1) \\ &\geq \min_{1 \leq i < k} \{\ell(n, i) + \ell(n, k - i)\}, \end{aligned}$$

where we used the monotonicity of $\ell(n, k)$ with respect to k for the second estimation.

The other case is that $E = F \cdot G$. We may assume that the words in F have length at most n – otherwise, we apply the argument to $E^R = G^R \cdot F^R$, and note that $\text{awidth}(E) = \text{awidth}(E^R)$. Let n_1 denote the length of the words in F . Then we have $1 \leq n_1 \leq n$. We claim that $L(F)$ must be a singleton language, that is, $L(F) = \{w\}$ for some word w . For the sake of contradiction, assume $L(F)$ contains another word x with $x \neq w$. Since E describes P_n and $L(E) = L(F) \cdot L(G)$, the language $L(E)$ contains a word of the form wzw^R , for some infix z . Since there is only one way to write wzw^R as product of words in $L(F)$ and $L(G)$, the word zw^R must be in $L(G)$. But then the non-palindromic word xzw^R is a member of $L(E)$, which yields the desired contradiction to establish the claim.

52:4 Optimal Regular Expressions for Palindromes of Given Length

Further, we can assume that $n_1 = 1$ without loss of generality. This can be seen as follows. By Lemma 1, $\text{awidth}(L(E)) = \text{awidth}(L(F)) + \text{awidth}(L(G))$. For $1 \leq i \leq n_1$, let a_i denote the i th letter in w . Since $L(F) = \{w\}$, we have $\text{awidth}(L(F)) = n_1$. Thus, if we replace the subexpression F of E with the expression $\tilde{f} = a_1 \cdot (a_2 \cdots a_{n_1})$, the expression \tilde{E} thus obtained is again minimal. By applying the associative law for concatenation to \tilde{E} , we obtain yet another minimal expression $\tilde{E}' = \tilde{f}' \cdot G''$ with $\tilde{f}' = a_1$ and $G'' = (a_2 \cdots a_{n_1} \cdot G)$.

Since all words in P_n are palindromic, $L(G'') = S \cdot a_1$, for some subset S of P_{n-2} . Also, set S must be of the same cardinality as $L(E)$. We thus obtain the following recurrence in the case of concatenation:

$$\ell(n, k) \geq \ell(n-1, k) + 2.$$

Observe that k can be at most 2^{n-1} in this case, since there are no more than 2^{n-1} palindromes of length $2(n-1)$. Also, we must have $n \geq 2$ in the case of concatenation, since both $k \geq 2$ and $k \leq 2^{n-1}$ hold.

Either the case of union or of concatenation applies – because E has no Kleene star, and we obtain the recurrence relation

$$\ell(n, k) \geq \min\{\ell(n-1, k) + 2, \min_{1 \leq i < k} \{\ell(n, i) + \ell(n, k-i)\}\}, \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

and

$$\ell(n, k) \geq \min_{1 \leq i < k} \{\ell(n, i) + \ell(n, k-i)\}, \text{ for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n,$$

as desired. ◀

In the above proof, we derived a recursive lower bound on $\ell(n, k)$. Let f denote the integer-valued function which is defined by that recurrence, that is,

$$f(n, k) = \min\{f(n-1, k) + 2, \min_{1 \leq i < k} \{f(n, i) + f(n, k-i)\}\}, \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

$$f(n, k) = \min_{1 \leq i < k} \{f(n, i) + f(n, k-i)\}, \text{ for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n,$$

and

$$f(n, 1) = 2n.$$

The recursive definition can be simplified with the aid of the following lemma.

► **Lemma 3.** $f(n, k) = f(n-1, k) + 2$, for $n \geq 2$ and $2 \leq k \leq 2^{n-1}$.

Proof. Recall the recursive definition of f in this parameter range is

$$f(n, k) = \min\{f(n-1, k) + 2, \min_{1 \leq i < k} \{f(n, i) + f(n, k-i)\}\},$$

for $n \geq 2$ and $2 \leq k \leq 2^{n-1}$, so the inequality $f(n, k) \leq f(n-1, k) + 2$ is immediate. For the converse inequality, we claim that

$$\min_{1 \leq i < k} \{f(n, i) + f(n, k-i)\} \geq f(n-1, k) + 2.$$

We prove this by lexicographic induction on (n, k) . To show the statement for $n \geq 2$ and $k \geq 2$, we assume that the statement holds for all pairs (n', k') with $n' < n$, as well as for all pairs with $n' = n$ and $k' < k$. Observe, that by the induction hypothesis on (n, k) we also can safely assume that $f(n', k') \geq f(n'-1, k') + 2$, which follows from the recursive definition of f . The base case $(2, 2)$ is easily verified with

$$\min_{1 \leq i < 2} \{f(2, i) + f(2, 2-i)\} = f(2, 1) + f(2, 1) = 4 + 4 \geq 4 + 2 = f(1, 2) + 2,$$

because $f(1, 2) = \min_{1 \leq i < 2} \{f(1, i) + f(1, 2 - i)\} = f(1, 1) + f(1, 1) = 2 + 2 = 4$. For the induction step, we apply the induction hypothesis and $f(n, k) \geq f(n - 1, k) + 2$ twice to obtain

$$\begin{aligned} \min_{1 \leq i < k} \{f(n, i) + f(n, k - i)\} &\geq \min_{1 \leq i < k} \{(f(n - 1, i) + 2) + (f(n - 1, k - i) + 2)\} \\ &\geq \min_{1 \leq i < k} \{f(n - 1, i) + f(n - 1, k - i)\} + 4 \\ &\geq f(n - 1, k) + 6, \end{aligned}$$

which means that $\min_{1 \leq i < k} \{f(n, i) + f(n, k - i)\} \geq f(n - 1, k) + 2$ as desired.

Having established the claim, the equality of $f(n, k)$ with $f(n - 1, k) + 2$ now follows immediately. This completes the proof of the lemma. ◀

Still, the second recurrence equation entails the full history of the parameter k . One might hope that f is convex in the parameter k , and that the minimum in the formula $\min_{1 \leq i < k} \{f(n, i) + f(n, k - i)\}$ is always attained in the middle, i.e., $\arg \min i = \lfloor \frac{k}{2} \rfloor$. Compare, e.g., [21, p. 366] on convex recurrences. But this is, unfortunately, not the case: for instance, we have $f(3, 6) = \min_{1 \leq i < 6} \{f(3, i) + f(3, 6 - i)\} = f(3, 2) + f(3, 4) = 8 + 14 = 22$, while $2 \cdot f(3, 3) = 2 \cdot (f(2, 3) + 2) = 2 \cdot (10 + 2) = 24$. In fact, computations for small ranges of n and k suggest a nontrivial behavior of f – see Table 1.

■ **Table 1** Some $f(n, k)$ values for small n and k .

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$	$k = 11$	$k = 12$	$k = 13$	$k = 14$	$k = 15$	$k = 16$
$n = 1$	2	4														
$n = 2$	4	6	10	12												
$n = 3$	6	8	12	14	20	22	26	28								
$n = 4$	8	10	14	16	22	24	28	30	38	40	44	46	52	54	58	60

At least, we are interested only in the value of $f(n, 2^n)$. Once we put forward a suitable induction hypothesis (which admittedly is somewhat flabbergasting), we can establish a simple closed form for $f(n, 2^n)$ with a laborious lexicographic induction.

▶ **Lemma 4.** $f(n, 2^n) = 2^{n+2} - 4$.

Proof. For the upper bound, observe that $f(n, 2^n) \leq 2 \cdot f(n, 2^{n-1})$ easily follows from the recurrence equations defining f and with the help of Lemma 3 we obtain

$$f(n, 2^n) \leq 2 \cdot (f(n - 1, 2^{n-1}) + 2).$$

With $f(1, 2) = 4$, this boils down to an inhomogeneous linear recurrence with variable n , which can be solved as $f(n, 2^n) \leq 4(2^n - 1) = 2^{n+2} - 4$.

The lower bound will follow immediately once we have established the following claim.

▷ **Claim 5.** Let $n \geq 1$ and $1 \leq k \leq 2^n$. Then

$$f(n, k) \geq \begin{cases} 4k & \text{if } k < 2^{n-1}, \\ 4k - 2 & \text{if } k \text{ is not a power of two and } k > 2^{n-1}, \text{ and} \\ 4k - 4 + 2n - 2 \log k & \text{if } k \text{ is a power of two.} \end{cases}$$

The remaining part of the proof is a lexicographic induction on (n, k) , which tedious details are left to the reader. ◀

4 Some Digit Theory

Now that our appetite is whetted, we want to solve the recurrence also in the general case where k is not a power of two. In this section, we develop the necessary tools regarding “digit theory,” that is, mathematical properties of digit sums, that we will need for the analysis. Let $S_2(n)$ denotes the “digit sum to base 2” function. This function is often referred to as the *Hamming weight function* and denotes the number of ones in the binary expansion of the number n . Throughout the rest of this paper, for a nonnegative integer n , we refer to the function $\lambda(n)$ defined as

$$\lambda(n) = \begin{cases} 0, & \text{if } n = 0 \\ \lfloor \log_2 n \rfloor, & \text{otherwise.} \end{cases}$$

Here $\log_2 n$ refers to the logarithm to base 2. For the digit sum to base 2 we find the following equations useful whenever powers of 2 are involved somehow.

► **Lemma 6.** *Let n be a nonnegative integer. Then*

1. $S_2(2^n - 1) = n$ and
2. $S_2(n - 2^{\lambda(n)}) = S_2(n) - 1$.

Next we recall an alternative characterization of the digit sum to base 2 that proves useful in the forthcoming calculations.

► **Lemma 7.** *Let n be a nonnegative integer. Then*

$$S_2(n) = n - \sum_{i=1}^{\infty} \left\lfloor \frac{n}{2^i} \right\rfloor.$$

Observe, that the sum contains only a finite number of non-zero summands.

More generally, for prime q the sum $\sum_{i=1}^{\infty} \left\lfloor \frac{n}{q^i} \right\rfloor$ is famously known to be equal to the largest power of q that divides $n!$ (Legendre’s formula [22]). For non-prime q , the latter equality ceases to hold in general, because for $n = 8$ and $q = 4$ the largest integer power of 4 that divides $8!$ is 3, because $8! = (2 \cdot 4) \cdot 7 \cdot (2 \cdot 3) \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 7 \cdot 5 \cdot 4^3 \cdot 3^2 \cdot 2$, while the sum evaluates to 2.

The study of the following maximization problem

$$\max_{0 \leq i \leq n} \{S_2(i) + S_2(n - i)\}$$

is essential for our main result. Remarkably, the formula on the right-hand side of the identity in Lemma 8 below is famously known as the number of multiplications to compute the $(n + 1)$ th power by the ancient Indian *Chandah-sutra method*. This appears as sequence A014701 in the On-line Encyclopedia of Integer Sequences, and is referred to as the *left-to-right binary method*³ in [20, Chap. 4.6.3].

► **Lemma 8.** *Let n be a nonnegative integer. Then*

$$\max_{0 \leq i \leq n} \{S_2(i) + S_2(n - i)\} = \lambda(n + 1) + S_2(n + 1) - 1.$$

³ We note that the formula given in [20, p. 463] refers to the *right-to-left binary method*. As explained there, the latter takes one more multiplication than the left-to-right binary method.

Proof. Observe, that $S_2(n)$ denotes the Hamming weight of n , that is, the number of ones in the binary expansion of the number n . We shall prove first the easier inequality, namely $\max_{0 \leq i \leq n} \{S_2(i) + S_2(n - i)\} \geq \lambda(n + 1) + S_2(n + 1) - 1$. It suffices to find a suitable decomposition $n = j + (n - j)$ for some j , which attains the bound. We choose $j = n + 1 - 2^{\lambda(n+1)}$. Then j is equal to $n + 1$ modulo $2^{\lambda(n+1)}$, and thus their binary expansions differ only in the highest order bit. In other words, $S_2(j) = S_2(n + 1) - 1$ by Lemma 6.2. Also, by the finite geometric series expansion,

$$n - j = n - \left(n + 1 - 2^{\lambda(n+1)}\right) = 2^{\lambda(n+1)} - 1 = \sum_{i=0}^{\lambda(n+1)-1} 1 \cdot 2^i,$$

and thus $S_2(n - j) = \lambda(n + 1)$ – see Lemma 6.1.

The converse inequality requires more effort, namely to prove that

$$\max_{0 \leq i \leq n} \{S_2(i) + S_2(n - i)\} \leq \lambda(n + 1) + S_2(n + 1) - 1.$$

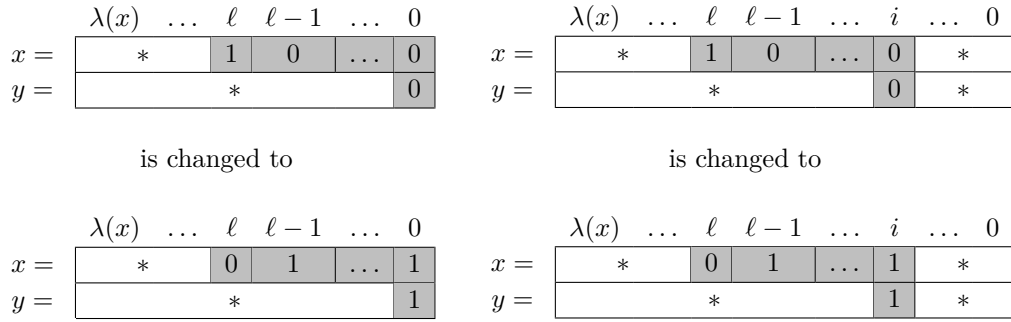
Our strategy is as follows. Given any decomposition $n = x + y$ with $x, y \geq 0$, we write x and y in binary positional notation $x_{\lambda(x)} \cdots x_1 x_0$ and $y_{\lambda(y)} \cdots y_1 y_0$. Then we shall apply a certain set of rules to x and y such that

- after each rule application, the sum of the two summands x' and y' thus obtained is n , that is, $x' + y' = n$,
- after each rule application, the sum of their Hamming weights is not decreased, that is, $S_2(x') + S_2(y') \geq S_2(x) + S_2(y)$, and
- after the last rule application, in the larger summand thus obtained, all bits are equal to 1.

This will of course suffice to show that the decomposition into j and $n - j$, as described at the beginning of the proof of this lemma, attains the maximum.

When looking at the bits of x and y , there are several constellations that need to be addressed. The first rule concerns the case $x_0 = y_0 = 0$, that is, the lowest order bits are both zero. Assume x is greater than or equal to y , otherwise we exchange the roles of x and y . Let ℓ denote the lowest order nonzero bit position of x , that is $x_\ell = 1$ and $x_k = 0$, for all k with $0 \leq k < \ell$. Then decreasing the number x by 1 amounts to setting $x_\ell = 0$ and $x_k = 1$, for all k with $0 \leq k < \ell$. Also, increasing the number y by 1 amounts to setting $y_0 = 1$, while all other bits of y remain unchanged. Observe, that this maneuver increases the Hamming weight of both summands. Also, the two summands thus obtained add up to n , and both summands have their lowest order bit set to 1. For an illustration of this situation we refer the reader to the left drawing of Figure 1.

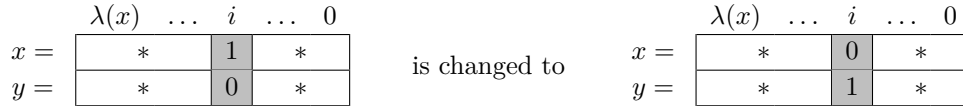
We now generalize this to the case $x_i = y_i = 0$, for $0 \leq i < \lambda(x)$. Here again, we assume that $x \geq y$; otherwise we exchange the roles of x and y . Here essentially the same mechanism applies, but, roughly speaking, we need to “multiply everything” by 2^i . In the same spirit as above, let ℓ denote the lowest order nonzero bit position of x above i , that is $x_\ell = 1$ and $x_k = 0$, for all k with $i \leq k < \ell$. Note that ℓ is guaranteed to exist, since $x \geq y$. Observe, that decreasing the number x by 2^i amounts to setting $x_\ell = 0$ and $x_k = 1$, for all k with $i \leq k < \ell$. Also, increasing the number y by 2^i amounts to setting $y_i = 1$, while all other bits of y remain unchanged. So this maneuver increases the Hamming weight of both summands. Also, the obtained summands sum up to n , and both summands have their i th bit set to 1. This completes the description of the first rule. For an illustration of this situation we refer to the right drawing of Figure 1.



■ **Figure 1** First bit manipulation rule for the decomposition of n into x and y for the first situation (left), i.e., $x_0 = y_0 = 0$, and the general situation (right), i.e., $x_i = y_i = 0$, for $0 \leq i < \lambda(x)$.

We iteratively apply this rule to the resulting pair of summands from the previous round, for each i in increasing order, requiring that $x \geq y$ at the beginning at every round; otherwise the rôles of x and y are exchanged. After the $(i + 1)$ th round, no constellations remain with $x_r = y_r = 0$, for $0 \leq r \leq i$. Finally, for every i with $0 \leq i \leq \lambda(x)$, no constellations remain with $x_i = y_i = 0$.

When y denotes the smaller summand obtained by the above procedure, the constellations where $y_i = 1$ do not need to be fixed. The remaining constellations are those where $y_i = 0$ and $x_i = 1$, for some $i \leq \lambda(x)$. The second rule is to exchange the bit values, that is, we set $y_i = 1$ and $x_i = 0$. It is clear that the two summands thus obtained add up to n . Also, the sum of the Hamming weights is unaffected. We apply the second rule as often as needed, and the number of these rule applications is of course bounded by $\lambda(x) + 1$. For an illustration of the second rule we refer to Figure 2.



■ **Figure 2** Second bit manipulation rule for the decomposition of n into x and y that is applied as often as needed.

After all applications of the second rule, we end up with the larger summand having all bits set to 1. Since the other two conditions are invariant under application of both rules, this completes the proof. ◀

5 Optimal Expressions for the first k Palindromes in Lexicographic Order

Now that we have collected the necessary tools, we aim to solve the recurrence $f(n, k)$ also for the case where k is not a power of two. Recall that Lemma 3 allows us to write up the recurrence in simplified form, as follows:

$$f(n, k) = f(n - 1, k) + 2, \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

$$f(n, k) = \min_{1 \leq i < k} \{f(n, i) + f(n, k - i)\}, \text{ for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n,$$

and

$$f(n, 1) = 2n.$$

We transform this recurrence into a recurrence on one unknown within two steps. In the first step, we define another function in two unknowns in terms of $f(n, k)$.

► **Lemma 9.** *Let $g(n, k) := \frac{1}{2}f(n, k) - n$. Then $g(n, k)$ satisfies the recurrence*

$$g(n, k) = g(n - 1, k), \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

$$g(n, k) = n + \min_{1 \leq i < k} \{g(n, i) + g(n, k - i)\}, \text{ for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n,$$

and

$$g(n, 1) = 0.$$

We shall apply the second transformation only for the “interesting” parameter range of k , that is, when k is in the upper half of the admissible range. Namely, we observe, that whenever $2^{n-1} < k \leq 2^n$, then we can express n in terms of k as $n = 1 + \lambda(k - 1)$. Recalling that $\lambda(0) = 0$, we set

$$h(k) := g(1 + \lambda(k - 1), k), \text{ for } k \geq 1.$$

Then we find the following situation:

► **Lemma 10.** *Let $h(k) := g(1 + \lambda(k - 1), k)$, for $k \geq 1$. Then $h(k)$ satisfies the recurrence*

$$h(1) = 0$$

$$h(k) = 1 + \lambda(k - 1) + \min_{1 \leq i < k} \{h(i) + h(k - i)\} \text{ for } k \geq 2,$$

and it has the solution $h(k) = 2(k - 1) - S_2(k - 1)$.

It is worth mentioning that the formula in Lemma 10 implies that the values of the recurrence h , starting from $h(1)$, coincide with the (zero-based) sequence A005187 in the On-Line Encyclopedia of Integer Sequences – the numbers having a partition into distinct Mersenne numbers greater than zero.

Now let’s undo both transformations. We first determine $g(n, k)$ by using $h(k)$ and its explicit solution. Then, by elementary calculations we arrive at an alternative recurrence for $f(n, k)$.

► **Lemma 11.** *The function $f(n, k)$ satisfies the recurrence*

$$f(n, k) = f(n - 1, k) + 2, \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

$$f(n, k) = 2n + 4(k - 1) - 2S_2(k - 1), \text{ for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n,$$

and

$$f(n, 1) = 2n$$

and it has the solution $f(n, k) = 2n + 4(k - 1) - 2S_2(k - 1)$.

Proof. In order to undo both transformations, we first determine $g(n, k)$ by using $h(k)$ and its explicit solution from Lemma 10. We find

$$g(n, k) = g(n - 1, k), \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

$$g(n, k) = 2(k - 1) - S_2(k - 1), \text{ for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n,$$

and

$$g(n, 1) = 0,$$

52:10 Optimal Regular Expressions for Palindromes of Given Length

because, for $k > 2^{n-1}$ we have

$$\begin{aligned} g(n, k) &= g(1 + \lambda(k - 1), k) \\ &= h(k) \\ &= 2(k - 1) - S_2(k - 1). \end{aligned}$$

Finally, recall that $f(n, k) = 2(g(n, k) + n)$, which results in

$$f(n, k) = f(n - 1, k) + 2, \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1},$$

and for $k > 2^{n-1}$ we calculate

$$\begin{aligned} f(n, k) &= 2(g(n, k) + n) \\ &= 2(2(k - 1) - S_2(k - 1) + n) \\ &= 2n + 4(k - 1) - 2S_2(k - 1). \end{aligned}$$

For the terminating cases of the recurrence, we simply recall those from the original recurrence defining $f(n, k)$:

$$f(n, 1) = 2n,$$

and this completes the proof.

It remains to solve the alternative recurrence, which is now done with ease. The statement is proved by lexicographic induction on (k, n) . Let $k = 1$, then $2n + 4(k - 1) - 2S_2(k - 1) = 2n$ is obviously an solution for any n . To show the statement for $k \geq 2$ and $n \geq 1$, we assume that the statement holds for all pairs (k', n') with $k' < k$, as well as for all pairs with $k' = k$ and $n' < n$. Then for the case $k > 2^{n-1}$ we have nothing to prove and in case $2 \leq k \leq 2^{n-1}$, we apply the induction hypothesis and get

$$\begin{aligned} f(n, k) &= f(n - 1, k) + 2 \\ &= 2(n - 1) + 4(k - 1) - 2S_2(k - 1) + 2 \\ &= 2n + 4(k - 1) - 2S_2(k - 1) \end{aligned}$$

as desired. ◀

With the lower bound in place, it remains to give an optimal regular expression matching the lower bound. The expression $E_{n,k}$ describes the lexicographically first k palindromes of length $2n$, and is defined recursively as follows:

$$\begin{aligned} E_{n,k} &= a \cdot E_{n-1,k} \cdot a, \text{ for } n \geq 1 \text{ and } 1 \leq k \leq 2^{n-1}, \\ E_{n,k} &= a \cdot E_{n-1,2^{n-1}} \cdot a + b \cdot E_{n-1,k-2^{n-1}} \cdot b, \text{ for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n, \end{aligned}$$

and

$$E_{0,1} = \epsilon.$$

We can prove by induction that this recursive upper bound on the alphabetic width meets the lower bound:

► **Lemma 12.** For $n \geq 0$ and $k \geq 1$, $\text{awidth}(E_{n,k}) = f(n, k)$.

It remains to show that the definition of $E_{n,k}$ is semantically correct, in the sense that it describes exactly the set of the lexicographically first k palindromes.

► **Lemma 13.** *Let n, k be integers with $n \geq 0$ and $1 \leq k \leq 2^n$. Then the regular expression $E_{n,k}$ describes the lexicographically first k palindromes of length $2n$.*

Proof. We begin with a natural bijection between palindromes of length $2n$, for $n \geq 1$, and the nonnegative integers in the range $0, 1, \dots, 2^n - 1$: for a nonnegative integer j with $0 \leq j < 2^n$, with binary expansion $\sum_{r=0}^{\infty} j_r 2^r = k$, let $\rho : \{0 \mapsto a, 1 \mapsto b\}$, and let $\rho_n(j) = \rho(j_{n-1})\rho(j_{n-2}) \cdots \rho(j_0)$ denote the usual n -bit binary representation of j in positional notation – with leading zeros if needed. Define the family of functions σ_n by letting $\sigma_n(j) = \rho_n(j)\rho_n(j)^R$. Whenever n is understood from the context, we shall drop the subscript and write $\rho(j)$ instead of $\rho_n(j)$, and similarly for σ . Observe, that, among the palindromes of length $2n$, the word $\sigma(j)$ is the $(j+1)$ th palindrome in lexicographic order. Conversely, for a palindrome w of length $2n$, the preimage $\sigma^{-1}(w)$ equals the (zero-based) lexicographic index of w among the palindromes of length $2n$. For convenience, we extend the definition of σ_n to the case $n = 0$ by letting $\sigma_0(0) = \epsilon$.

We claim that, given $n \geq 1$ and k with $1 \leq k \leq 2^n$, as well as a nonnegative integer $j < k$, the word $\sigma(j)$ is in $L(E_{n,k})$. This claim will be proven by induction on n . The base case is $n = 0$. We thus have $k = 1$. Here, $\sigma_0(0) = \epsilon$, and $E_{0,1} = \epsilon$. For the induction step, we now assume $n \geq 1$. We consider two cases:

Case 1. Consider first the case $j < 2^{n-1}$. Then $\sigma_n(j) = a \cdot \sigma_{n-1}(j) \cdot a$. By the induction hypothesis, $\sigma_{n-1}(j) \in L(E_{n-1,k})$, and by the recursive definition of the regular expression $E_{n,k}$, we have $a \cdot L(E_{n-1,k}) \cdot a \subseteq L(E_{n,k})$. Hence, $\sigma(j) \in L(E_{n,k})$ in this case.

Case 2. The other case is $j \geq 2^{n-1}$. Then $\sigma_n(j) = b \cdot \sigma_{n-1}(j - 2^{n-1}) \cdot b$. Observe, that also $k > 2^{n-1}$ holds, since $j < k$. Let $k' = k - 2^{n-1}$ and $j' = j - 2^{n-1}$. Then $k' \geq 1$ and $j' \geq 0$, as well as $n - 1 \geq 0$. Using the induction hypothesis, we have $\sigma_{n-1}(j') \in L(E_{n-1,k'})$. In other words, $\sigma_{n-1}(j - 2^{n-1}) \in L(E_{n-1,k-2^{n-1}})$. Now, by the recursive definition of the regular expression $E_{n,k}$, we obtain $b \cdot L(E_{n-1,k-2^{n-1}}) \cdot b \subseteq L(E_{n,k})$. Hence, $\sigma(j) \in L(E_{n,k})$ also in this case.

This completes the induction, and the claim is established.

It remains to show that no other words are described by $E_{n,k}$. To this end, we note first that the recursive definition of $E_{n,k}$ ensures that it describes no non-palindromic words, and only words of length $2n$. Now let w be any word that is described by $E_{n,k}$. Recall that $\sigma^{-1}(w)$ is equal to the lexicographic index of w among all palindromes of length $2n$.

We shall prove by induction on n that the lexicographic index of every w described by $E_{n,k}$ is at most $k - 1$. In the base case $n = 0$, we must have $k = 1$ and $w = \epsilon$, and $\sigma^{-1}(w) = 0 = k - 1$ in this case. Now assume $n \geq 1$. We distinguish two cases:

Case 1. Consider first the case that the word w is the form axa . We need to consider two subcases. The first subcase is $k \leq 2^{n-1}$. Here, by definition of $E_{n,k}$, the word x is in $L(E_{n-1,k})$. Bearing in mind that $\sigma^{-1}(x) = \sigma_{n-1}^{-1}(x)$ and $\sigma^{-1}(w) = \sigma_n^{-1}(w)$ denote two different functions, we will again drop the subscripts for convenient reading. By the induction assumption, $\sigma^{-1}(x) \leq k - 1$. Recalling the bijection between natural numbers and palindromes, we have $\sigma^{-1}(axa) = \sigma^{-1}(x)$. With $axa = w$, we obtain $\sigma^{-1}(w) = \sigma^{-1}(x)$ in this subcase. The second subcase is $k > 2^{n-1}$. Here, by definition of $E_{n,k}$, the word x is in $L(E_{n-1,2^{n-1}})$. By the induction assumption, $\sigma^{-1}(x) \leq 2^{n-1} \leq k - 1$, and using again the bijection between natural numbers and palindromes, $\sigma^{-1}(w) = \sigma^{-1}(x)$.

Case 2. Now consider the case that the word w is of the form $bx b$. By the recursive definition of $E_{n,k}$, we can conclude that $x \in L(E_{n-1,k-2^{n-1}})$ – and that $k > 2^{n-1}$. By the induction assumption, $\sigma^{-1}(x) \leq k - 2^{n-1} - 1$. Recalling the bijection between natural numbers and palindromes, we have $\sigma^{-1}(bx b) = 2^{n-1} + \sigma^{-1}(x)$. Taking these two facts together, we obtain $\sigma^{-1}(w) = 2^{n-1} + \sigma^{-1}(x) \leq 2^{n-1} + k - 2^{n-1} - 1 \leq k - 1$, as desired.

This completes the proof of the second claim, and the proof of the lemma is completed. ◀

52:12 Optimal Regular Expressions for Palindromes of Given Length

We thus can summarize our findings about palindromes of even length in the last three lemmata in the following statement.

► **Theorem 14.** *Let k and n be integers, with $n \geq 0$ and $1 \leq k \leq 2^n$. Then the set of the lexicographically first k palindromes of length $2n$ over a binary alphabet requires regular expressions of alphabetic width exactly $2n + 4(k - 1) - 2S_2(k - 1)$.*

6 Alphabetic Width of Palindromes of Odd Length

We turn to palindromes of odd length. The recurrences essentially differ only in the terminating cases. But changing the starting conditions of a nonlinear system may, or may not, change everything. We thus provide a careful writeup.

To this end, for positive integer n , let \tilde{P}_n denote the set of palindromes of length $2n - 1$ over a binary alphabet. Now define $\tilde{\ell}(n, k)$ to be the minimum alphabetic width of a regular expression describing a subset of \tilde{P}_n , where the subset has cardinality at least k . Again by definition, $\tilde{\ell}(n, k)$ is monotone with respect to the parameter k .

► **Lemma 15.** *Let $n \geq 1$ and $1 \leq k \leq 2^n$. Then $\tilde{\ell}(n, k)$ obeys the following recurrence:*

$$\begin{aligned} \tilde{\ell}(n, k) &\geq \min\{\tilde{\ell}(n-1, k) + 2, \min_{1 \leq i < k} \{\tilde{\ell}(n, i) + \tilde{\ell}(n, k-i)\}\}, \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1}, \\ \tilde{\ell}(n, k) &\geq \min_{1 \leq i < k} \{\tilde{\ell}(n, i) + \tilde{\ell}(n, k-i)\}, \text{ for } n \geq 1 \text{ and } k > 2^{n-1}, \end{aligned}$$

and

$$\tilde{\ell}(n, 1) = 2n - 1.$$

In analogy to the definition of the function f , let \tilde{f} denote the integer-valued function which is defined by that recurrence, that is,

$$\begin{aligned} \tilde{f}(n, k) &= \min\{\tilde{f}(n-1, k) + 2, \min_{1 \leq i < k} \{\tilde{f}(n, i) + \tilde{f}(n, k-i)\}\}, \text{ for } n \geq 2 \text{ and } 2 \leq k \leq 2^{n-1}, \\ \tilde{f}(n, k) &= \min_{1 \leq i < k} \{\tilde{f}(n, i) + \tilde{f}(n, k-i)\}, \text{ for } n \geq 1 \text{ and } 2^{n-1} < k \leq 2^n, \end{aligned}$$

and

$$\tilde{f}(n, 1) = 2n - 1.$$

We estimate the values of the function $\tilde{f}(n, k)$ as follows:

► **Lemma 16.** *Let $n \geq 1$ and $1 \leq k \leq 2^n$. Then $\tilde{f}(n, k) = f(n, k) - k$.*

Thus, we immediately obtain:

► **Lemma 17.** $\tilde{f}(n, 2^n) = 3 \cdot 2^n - 4$.

With the lower bound in place, it remains to give an optimal regular expression matching the lower bound. The expression $\tilde{E}_{n,k}$ is defined recursively as follows.

$$\begin{aligned} \tilde{E}_{n,k} &= a \cdot \tilde{E}_{n-1,k} \cdot a, \text{ for } n \geq 2 \text{ and } 1 \leq k \leq 2^{n-1}, \\ \tilde{E}_{n,k} &= a \cdot \tilde{E}_{n-1,2^{n-1}} \cdot a + b \cdot \tilde{E}_{n-1,k-2^{n-1}} \cdot b, \text{ for } n \geq 2 \text{ and } 2^{n-1} < k \leq 2^n, \end{aligned}$$

and

$$\tilde{E}_{1,1} = a \text{ as well as } \tilde{E}_{1,2} = a + b.$$

The semantic correctness proof runs along the lines of the proof of Lemma 13.

► **Lemma 18.** *Let n, k be integers with $n \geq 1$ and $1 \leq k \leq 2^n$. Then the regular expression $\tilde{E}_{n,k}$ describes the lexicographically first k palindromes of length $2n - 1$.*

It remains to show that the alphabetic width of $\tilde{E}_{n,k}$ meets the lower bound. An easy induction reduces this to the case of even length palindromes, in a similar vein as we did it in Lemma 16.

► **Lemma 19.** *Let n, k be integers with $n \geq 1$ and $1 \leq k \leq 2^n$. Then $\text{awidth}(\tilde{E}_{n,k}) = \text{awidth}(E_{n,k}) - k$.*

We thus can summarize our findings about palindromes of odd length in the following theorem – compare with Theorem 14.

► **Theorem 20.** *Let k and n be integers, with $n \geq 1$ and $1 \leq k \leq 2^n$. Then the set of the lexicographically first k palindromes of length $2n - 1$ over a binary alphabet requires regular expressions of alphabetic width exactly $2n + 3(k - 1) - 2S_2(k - 1) - 1$.*

We conclude this section with a curious observation, which was contributed by an anonymous reviewer. Recall that

$$\tilde{\ell}(n, k) = \min_{\substack{|L| \geq k \\ L \subseteq \tilde{P}_n}} \{\text{awidth}(L)\},$$

that is, $\tilde{\ell}(n, k)$ denotes the minimum alphabetic width of a regular expression describing a subset of \tilde{P}_n , where the subset has cardinality at least k . Then the analysis in the present work establishes that the minimum is attained by the set of the lexicographically first k palindromes, and a corresponding statement holds in the even length case. This observation is summarized in the following theorem (which no longer needs to distinguish between even and odd length):

► **Theorem 21.** *For $n \geq 0$ and $1 \leq k \leq 2^{\lceil n/2 \rceil}$, let Pal_n denote the set of palindromes of length n , and let $\text{Lex}_{n,k}$ denote the set of the lexicographically first k palindromes of length n . Then*

$$\text{Lex}_{n,k} \in \underset{\substack{|L| \geq k \\ L \subseteq \text{Pal}_n}}{\text{argmin}} \text{awidth}(L).$$

As the reviewer pointed out, this is reminiscent of the Kruskal-Katona Theorem from extremal combinatorics, see, e.g., [19]. Among several equivalent formulations of that theorem, one of them deals with minimization of the size of shadows in layers of the Boolean hypercube. The Kruskal-Katona Theorem then states that initial segments with respect to a version of the lexicographic ordering form sets with the smallest shadow possible.

7 Conclusion

Most lower bound proofs for regular expression size can be put into the following three categories: proofs based on (arithmetic) circuit complexity, e.g., [4, 7, 14], proofs based on the star height lemma, e.g., [9, 12, 13], and specialized proofs that are tailor-made for a specific language family, e.g., [2, 6, 10, 23]. While the present work falls into the third category, the lower bound method is quite similar to the one for permutations [23]. We expect that the method can be expanded to further families of finite languages, where the best known regular expressions have a divide-and-conquer flavor. A few examples from the literature come to mind:

- First, the binomial language $B_{n,k} = \{w \in \{0,1\}^n : |w|_1 = k\}$. A regular expression of divide-and-conquer flavor having size $n^{O(\log k)}$ for this language was proposed in [7], and the question of optimality was posed as an open problem. In [4], methods from arithmetic circuit complexity are utilized to derive a lower bound of $nk^{\Omega(\log k)}$.
- Regarding larger alphabets, the less-than relation on an n -set is given as $\{ij \mid 1 \leq i < j \leq n\}$. For this language, the minimum required regular expression size was determined exactly in [2], which implies a lower bound on the complexity of rectifier networks. This language naturally generalizes to the set of increasing sequences of length k over an n -set. For this an arithmetic formula lower bound was derived in [17]. As pointed out in [4], that result transfers to lower bounds on regular expression size.
- For the set of permutations of an n -set, the exact bound was determined in [23], and an asymptotic lower bound is given in [4] using a different method. Its natural generalization is the set of k -permutations of an n -set. The nondeterministic state complexity of this language is studied in [1]. Their motivation is that a lower bound on nondeterministic state complexity gives lower bounds on the running time for parameterized algorithms following the divide-and-conquer paradigm. We claim that, by the well-nested nature of divide-and-conquer, a (potentially higher) lower bound on regular expression size would serve this goal equally well.

The cited examples witness a lot of cross-fertilization between lower bound methods on various models of computation, including arithmetic circuits, rectifier networks, families of parameterized algorithms, and, of course, regular expressions.

References

- 1 R. Ben-Basat, A. Gabizon, and M. Zehavi. The k -distinct language: Parameterized automata constructions. *Theoretical Computer Science*, 622:1–15, 2016.
- 2 D. Chistikov, Sz. Iván, A. Lubiw, and J. Shallit. Fractional coverings, greedy coverings, and rectifier networks. In Heribert Vollmer and Brigitte Vallée, editors, *Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.
- 3 N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.
- 4 E. Cseresnyes and H. Seiwert. Regular expression length via arithmetic formula complexity. In G. Jirásková and G. Pighizzini, editors, *22nd International Conference on Descriptive Complexity of Formal Systems*, volume 12442 of *LNCS*, pages 26–38. Springer, 2020.
- 5 K. Edwards and G. Farr. Improved upper bounds for planarization and series-parallelization of degree-bounded graphs. *The Electronic Journal of Combinatorics*, 19(2):#P25, 2012.
- 6 A. Ehrenfeucht and H. P. Zeiger. Complexity measures for regular expressions. *Journal of Computer and System Sciences*, 12(2):134–146, April 1976. doi:10.1016/S0022-0000(76)80034-7.
- 7 K. Ellul, B. Krawetz, J. Shallit, and M.-W. Wang. Regular expressions: New results and open problems. *Journal of Automata, Languages and Combinatorics*, 10(4):407–437, 2005.
- 8 Y. Gao, N. Moreira, R. Reis, and S. Yu. A survey on operational state complexity. *Journal of Automata, Languages and Combinatorics*, 21(4):251–310, 2016.
- 9 W. Gelade. Succinctness of regular expressions with interleaving, intersection, and counting. *Theoretical Computer Science*, 411(31–33):2987–2998, 2010. doi:10.1016/j.tcs.2010.04.036.
- 10 W. Gelade and F. Neven. Succinctness of the complement and intersection of regular expressions. *ACM Transactions on Computational Logic*, 13(1):No. 4, January 2012. doi:10.1145/2071368.2071372.

- 11 D. H. Greene and D. E. Knuth. *Mathematics for the Analysis of Algorithms*. Progress in Computer Science. Birkhäuser, 2nd edition, 1982.
- 12 H. Gruber and M. Holzer. Finite automata, digraph connectivity, and regular expression size. In L. Aceto, I. Damgaard, L. A. Goldberg, M. M. Halldórsson, A. Ingólfssdóttir, and I. Walkuwiewicz, editors, *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, number 5126 in LNCS, pages 39–50, Reykjavik, Iceland, July 2008. Springer. doi:10.1007/978-3-540-70583-3_4.
- 13 H. Gruber and M. Holzer. Tight bounds on the descriptonal complexity of regular expressions. In V. Diekert and D. Nowotka, editors, *Proceedings of the 13th International Conference Developments in Language Theory*, number 5583 in LNCS, pages 276–287, Stuttgart, Germany, June–July 2009. Springer. doi:10.1007/978-3-642-02737-6_22.
- 14 H. Gruber and J. Johannsen. Tight bounds on the descriptonal complexity of regular expressions. In R. Amadio, editor, *Proceedings of the 11th Conference Foundations of Software Science and Computational Structures*, number 4962 in LNCS, pages 273–286, Budapest, Hungary, March–April 2008. Springer.
- 15 M. Holzer and M. Kutrib. Descriptonal and computational complexity of finite automata—a survey. *Information and Computation*, 209(3):456–470, March 2011. doi:10.1016/j.ic.2010.11.013.
- 16 J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- 17 P. Hrubec and A. Yehudayoff. Homogeneous formulas and symmetric polynomials. *Computational Complexity*, 20(3):559–578, 2011.
- 18 H.-K. Hwang and T.-H. Tsai. An asymptotic theory for recurrence relations based on minimization and maximization. *Theoretical Computer Science*, 290(3):1475–1501, 2003.
- 19 S. Jukna. *Extremal Combinatorics: With Applications in Computer Science*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2nd edition, 2011.
- 20 D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 3rd edition, 1998.
- 21 D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, 3rd edition, 1998.
- 22 A.-M. Legendre. *Essai sur la théorie des nombres*. Courcier, 2ème edition, 1808.
- 23 A. M. Lovett and J. O. Shallit. Optimal regular expressions for permutations. In Ch. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, editors, *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *LIPICs*, pages 121:1–121:12, Patras, Greece, July 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct

Shibashis Guha  

Tata Institute of Fundamental Research, Mumbai, India

Ismaël Jecker  

IST Austria, Klosterneuburg, Austria

Karoliina Lehtinen  

CNRS, Aix-Marseille University and University of Toulon, LIS, Marseille, France

Martin Zimmermann  

University of Liverpool, UK

Abstract

We study the expressiveness and succinctness of good-for-games pushdown automata (GFG-PDA) over finite words, that is, pushdown automata whose nondeterminism can be resolved based on the run constructed so far, but independently of the remainder of the input word.

We prove that GFG-PDA recognise more languages than deterministic PDA (DPDA) but not all context-free languages (CFL). This class is orthogonal to unambiguous CFL. We further show that GFG-PDA can be exponentially more succinct than DPDA, while PDA can be double-exponentially more succinct than GFG-PDA. We also study GFGness in visibly pushdown automata (VPA), which enjoy better closure properties than PDA, and for which we show GFGness to be EXPTIME-complete. GFG-VPA can be exponentially more succinct than deterministic VPA, while VPA can be exponentially more succinct than GFG-VPA. Both of these lower bounds are tight.

Finally, we study the complexity of resolving nondeterminism in GFG-PDA. Every GFG-PDA has a positional resolver, a function that resolves nondeterminism and that is only dependant on the current configuration. Pushdown transducers are sufficient to implement the resolvers of GFG-VPA, but not those of GFG-PDA. GFG-PDA with finite-state resolvers are determinisable.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Pushdown Automata, Good-for-games, Synthesis, Succinctness

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.53

Related Version *Full Version*: <https://arxiv.org/abs/2105.02611> [14]

Funding *Ismaël Jecker*: Funded by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754411.

Karoliina Lehtinen: Funded by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 892704.

1 Introduction

Nondeterminism adds both expressiveness and succinctness to deterministic pushdown automata. Indeed, the class of context-free languages (CFL), recognised by nondeterministic pushdown automata (PDA), is strictly larger than the class of deterministic context-free languages (DCFL), recognised by deterministic pushdown automata (DPDA), both over finite and infinite words. Even when restricted to languages in DCFL, there is no computable bound on the relative succinctness of PDA [15, 38]. In other words, nondeterminism is remarkably powerful, even for representing deterministic languages. The cost of such succinct representations is algorithmic: problems such as universality and solving games with a CFL winning condition are undecidable for PDA [11, 19], while they are decidable for DPDA [39].



© Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 53; pp. 53:1–53:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Intermediate forms of automata that lie between deterministic and nondeterministic models have the potential to mitigate some of the disadvantages of fully nondeterministic automata while retaining some of the benefits of the deterministic ones.

Unambiguity and bounded ambiguity, for example, restrict nondeterminism by requiring words to have at most one or at most k , for some fixed k , accepting runs. Holzer and Kutrib survey the noncomputable succinctness gaps between unambiguous PDA and both PDA and DPDA [18], while Okhotin and Salomaa show that unambiguous visibly pushdown automata are exponentially more succinct than DPDA [31]. Universality of unambiguous PDA is decidable, as it is decidable for unambiguous context-free grammars [33], which are effectively equivalent [17]. However, to the best of our knowledge, unambiguity is not known to reduce the algorithmic complexity of solving games with a context-free winning condition.

Another important type of restricted nondeterminism that is known to reduce the complexity of universality and solving games has been studied under the names of good-for-games (GFG) nondeterminism [16] and history-determinism [10]. Intuitively, a nondeterministic automaton is GFG if its nondeterminism can be resolved on-the-fly, i.e. without knowledge of the remainder of the input word to be processed.

For finite automata on finite words, where nondeterminism adds succinctness, but not expressiveness, GFG nondeterminism does not even add succinctness: every GFG-NFA contains an equivalent DFA [6], which can be obtained by pruning transitions from the GFG-NFA. Thus, GFG-NFA cannot be more succinct than DFA. But for finite automata on infinite words, where nondeterminism again only adds succinctness, but not expressiveness, GFG coBüchi automata can be exponentially more succinct than deterministic automata [23]. Finally, for certain quantitative automata over infinite words, GFG nondeterminism adds as much expressiveness as arbitrary nondeterminism [10].

Recently, pushdown automata on infinite words with GFG nondeterminism (ω -GFG-PDA) were shown to be strictly more expressive than ω -DPDA, while universality and solving games for ω -GFG-PDA are not harder than for ω -DPDA [25]. Thus, GFG nondeterminism adds expressiveness without increasing the complexity of these problems, i.e. pushdown automata with GFG nondeterminism induce a novel and intriguing class of context-free ω -languages.

Here, we continue this work by studying the expressiveness *and* succinctness of PDA over finite words. While the decidability results for ω -GFG-PDA on infinite words also hold for GFG-PDA on finite words, the separation argument between ω -GFG-PDA and ω -DPDA depends crucially on combining GFG nondeterminism with the coBüchi acceptance condition. Since this condition is only relevant for infinite words, the separation result does not transfer to the setting of finite words.

Nevertheless, we prove that GFG-PDA are more expressive than DPDA, yielding the first class of automata on finite words where GFG nondeterminism adds expressiveness. The language witnessing the separation is remarkably simple, in contrast to the relatively subtle argument for the infinitary result [25]: the language $\{a^i a^j b^k \mid k \leq \max(i, j)\}$ is recognised by a GFG-PDA but not by a DPDA. This yields a new class of languages, those recognised by GFG-PDA over finite words, for which universality and solving games are decidable. We also show that this class is incomparable with unambiguous context-free languages.

We then turn our attention to succinctness of GFG-PDA. We show that the succinctness gap between DPDA and GFG-PDA is at least exponential, while the gap between GFG-PDA and PDA is at least double-exponential. These results hold already for finite words.

To the best of our knowledge, both our expressiveness and our succinctness results are the first examples of good-for-games nondeterminism being used effectively over finite, rather than infinite, words (recall that all GFG-NFA are determinisable by pruning). Also, this

is the first succinctness result for good-for-games automata that does not depend on the infinitary coBüchi acceptance condition, which was used to show the exponential succinctness of GFG coBüchi automata, as compared to deterministic ones [23].

We then study an important subclass of GFG-PDA, namely, GFG visibly pushdown automata (VPA), in which the stack behaviour (push, pop, skip) is determined by the input letter only. GFG-VPA enjoy the good closure properties of VPA (to which they are expressively equivalent): they are closed under complement, union and intersection. We show that there is an exponential succinctness gap between deterministic VPA (DVPA) and GFG-VPA, as well as between GFG-VPA and VPA. Both of these are tight, as VPA, and therefore GFG-VPA as well, admit an exponential determinisation procedure [2]. Furthermore, we show that GFGness of VPA is decidable in EXPTIME. This makes GFG-VPA a particularly interesting class of PDA as they are recognisable, succinct, have good closure properties and deciding universality and solving games are both in EXPTIME. In contrast, solving ω -VPA games is 2EXPTIME-complete [27]. We also relate the problem of checking GFGness with the *good-enough synthesis* [1] or *uniformization* problem [9], which we show to be EXPTIME-complete for DVPA and GFG-PDA.

Nondeterminism in GFG automata is resolved on-the-fly, i.e. the next transition to be taken only depends on the run prefix constructed so far and the next letter to be processed. Thus, the complexity of a resolver, mapping run prefixes and letters to transitions, is a natural complexity measure for GFG automata. For example, finite GFG automata (on finite and infinite words) have a finite-state resolver [16]. For pushdown automata with their infinite configuration space, the situation is markedly different: On one hand, we show that GFG-PDA admit positional resolvers, that is, resolvers that depend only on the current configuration, rather than on the entire run prefix produced so far. Note that this result only holds for GFG-PDA over finite words, but not for ω -GFG-PDA. Yet, positionality does not imply that resolvers are simple to implement. We show that there are GFG-PDA that do not admit a resolver implementable by a pushdown transducer. In contrast, all GFG-VPA admit pushdown resolvers, again showing that GFG-VPA are better behaved than general GFG-PDA. Finally, GFG-PDA with finite-state resolvers are determinisable.

All proofs omitted due to space restrictions can be found in the full version [14].

Related work

The notion of GFG nondeterminism has emerged independently several times, at least as Colcombet's history-determinism [10], in Piterman and Henzinger's GFG automata [16], and as Kupferman, Safra, and Vardi's nondeterminism for recognising derived languages, that is, the language of trees of which all branches are in a regular language [24]. Related notions have also emerged in the context of XML document parsing. Indeed, preorder typed visibly pushdown languages and 1-pass preorder typeable tree languages, considered by Kumar, Madhusudan, and Viswanathan [21] and Martens, Neven, Schwentick, and Bex [28] respectively, also consider nondeterminism which can be resolved on-the-fly. However, the restrictions there are stronger than simple GFG nondeterminism, as they also require the typing to be unique, roughly corresponding to unambiguity in automata models and grammars. This motivates the further study of unambiguous GFG automata, although this remains out of scope for the present paper. The XML extension AXML has also inspired Active Context Free Games [29], in which one player, aiming to produce a word within a target regular language, chooses positions on a word and the other player chooses a rewriting rule from a context-free grammar. Restricting the strategies of the first player to moving from left to right makes finding the winner decidable [29, 5]; however, since the player still knows the future of the word, this restriction is not directly comparable to GFG nondeterminism.

Unambiguity, or bounded ambiguity, is an orthogonal way of restricting nondeterminism by limiting the number of permitted accepting runs per word. For regular languages, it leads to polynomial equivalence and containment algorithms [37]. Minimization remains NP-complete for both unambiguous automata [20, 4] and GFG automata [35] (at least when acceptance is defined on states, see [32]). On pushdown automata, increasing the permitted degree of ambiguity leads to both greater expressiveness and unbounded succinctness [17]. Finally, let us mention two more ways of measuring—and restricting—nondeterminism in PDA: bounded nondeterminism, as studied by Herzog [17] counts the branching in the run-tree of a word, while the minmax measure [34, 13] counts the number of nondeterministic guesses required to accept a word. The natural generalisation of GFGness as the *width* of an automaton [22] has not yet, to the best of our knowledge, been studied for PDA.

2 Preliminaries

An alphabet Σ is a finite nonempty set of letters. The empty word is denoted by ε , the length of a word w is denoted by $|w|$, and the n^{th} letter of w is denoted by $w(n)$ (starting with $n = 0$). The set of (finite) words over Σ is denoted by Σ^* , the set of nonempty (finite) words over Σ by Σ^+ , and the set of finite words of length at most n by $\Sigma^{\leq n}$. A language over Σ is a subset of Σ^* .

For alphabets Σ_1, Σ_2 , we extend functions $f: \Sigma_1 \rightarrow \Sigma_2^*$ homomorphically to words over Σ_1 via $f(w) = f(w(0))f(w(1))f(w(2)) \cdots$.

2.1 Pushdown automata

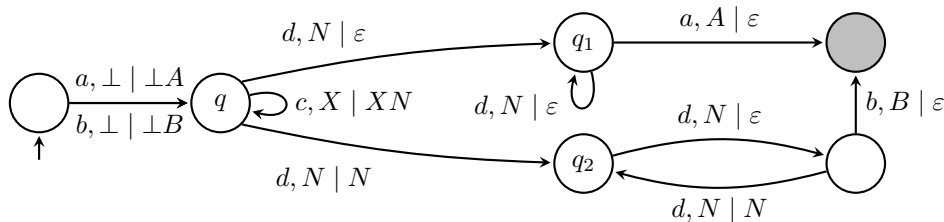
A pushdown automaton (PDA for short) $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ consists of a finite set Q of states with the initial state $q_I \in Q$, an input alphabet Σ , a stack alphabet Γ , a transition relation Δ to be specified, and a set F of final states. For notational convenience, we define $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ and $\Gamma_\perp = \Gamma \cup \{\perp\}$, where $\perp \notin \Gamma$ is a designated stack bottom symbol. Then, the transition relation Δ is a subset of $Q \times \Gamma_\perp \times \Sigma_\varepsilon \times Q \times \Gamma_\perp^{\leq 2}$ that we require to neither write nor delete the stack bottom symbol from the stack: If $(q, \perp, a, q', \gamma) \in \Delta$, then $\gamma \in \perp \cdot (\Gamma \cup \{\varepsilon\})$, and if $(q, X, a, q', \gamma) \in \Delta$ for $X \in \Gamma$, then $\gamma \in \Gamma^{\leq 2}$. Given a transition $\tau = (q, X, a, q', \gamma)$ let $\ell(\tau) = a \in \Sigma_\varepsilon$. We say that τ is an $\ell(\tau)$ -transition and that τ is a Σ -transition, if $\ell(\tau) \in \Sigma$. For a finite sequence ρ over Δ , the word $\ell(\rho) \in \Sigma^*$ is defined by applying ℓ homomorphically to every transition. We take the size of \mathcal{P} to be $|Q| + |\Gamma|$.¹

A stack content is a finite word in $\perp\Gamma^*$ (i.e. the top of the stack is at the end) and a configuration $c = (q, \gamma)$ of \mathcal{P} consists of a state $q \in Q$ and a stack content γ . The initial configuration is (q_I, \perp) .

The set of modes of \mathcal{P} is $Q \times \Gamma_\perp$. A mode (q, X) enables all transitions of the form (q, X, a, q', γ') for some $a \in \Sigma_\varepsilon$, $q' \in Q$, and $\gamma' \in \Gamma_\perp^{\leq 2}$. The mode of a configuration $c = (q, \gamma X)$ is (q, X) . A transition τ is enabled by c if it is enabled by c 's mode. In this case, we write $(q, \gamma X) \xrightarrow{\tau} (q', \gamma\gamma')$, where $\tau = (q, X, a, q', \gamma')$.

A run of \mathcal{P} is a finite sequence $\rho = c_0\tau_0c_1\tau_1 \cdots c_{n-1}\tau_{n-1}c_n$ of configurations and transitions with c_0 being the initial configuration and $c_{n'} \xrightarrow{\tau_{n'}} c_{n'+1}$ for every $n' < n$. The run ρ is a run of \mathcal{P} on $w \in \Sigma^*$, if $w = \ell(\rho)$. We say that ρ is accepting if it ends in a configuration whose state is final. The language $L(\mathcal{P})$ recognized by \mathcal{P} contains all $w \in \Sigma^*$ such that \mathcal{P} has an accepting run on w .

¹ Note that we prove exponential succinctness gaps, so the exact definition of the size is irrelevant, as long as it is polynomial in $|Q|$ and $|\Gamma|$. Here, we pick the sum for the sake of simplicity.



■ **Figure 1** The PDA \mathcal{P} from Example 2. Grey states are final, and X is an arbitrary stack symbol.

► **Remark 1.** Let $c_0\tau_0c_1\tau_1\cdots c_{n-1}\tau_{n-1}c_n$ be a run of \mathcal{P} . Then, the sequence $c_0c_1\cdots c_{n-1}c_n$ of configurations is uniquely determined by the sequence $\tau_0\tau_1\cdots\tau_{n-1}$ of transitions. Hence, whenever convenient, we treat a sequence of transitions as a run if it indeed induces one (not every such sequence does induce a run, e.g. if a transition $\tau_{n'}$ is not enabled in $c_{n'}$).

We say that a PDA \mathcal{P} is deterministic (DPDA) if

- every mode of \mathcal{P} enables at most one a -transition for every $a \in \Sigma \cup \{\varepsilon\}$, and
- for every mode of \mathcal{P} , if it enables some ε -transition, then it does not enable any Σ -transition.

Hence, for every input and for every run prefix on it there is at most one enabled transition to continue the run. Still, due to the existence of ε -transitions, a DPDA can have more than one run on a given input. However, these only differ by trailing ε -transitions.

The class of languages recognized by PDA is denoted by CFL, the class of languages recognized by DPDA by DCFL.

► **Example 2.** The PDA \mathcal{P} depicted in Figure 1 recognizes the language $\{ac^nd^na \mid n \geq 1\} \cup \{bc^nd^{2n}b \mid n \geq 1\}$. Note that while \mathcal{P} is nondeterministic, $L(\mathcal{P})$ is in DCFL.

2.2 Good-for-games Pushdown Automata

Here, we introduce good-for-games pushdown automata on finite words (GFG-PDA for short), nondeterministic pushdown automata whose nondeterminism can be resolved based on the run prefix constructed so far and on the next input letter to be processed, but independently of the continuation of the input beyond the next letter.

As an example, consider the PDA \mathcal{P} from Example 2. It is nondeterministic, but knowing whether the first transition of the run processed an a or a b allows the nondeterminism to be resolved in a configuration of the form $(q, \gamma N)$ when processing a d : in the former case, take the transition to state q_1 , in the latter case the transition to state q_2 . Afterwards, there are no nondeterministic choices to make and the resulting run is accepting whenever the input is in the language. This automaton is therefore good-for-games.

Formally, a PDA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ is good-for-games if there is a (nondeterminism) resolver for \mathcal{P} , a function $r: \Delta^* \times \Sigma \rightarrow \Delta$ such that for every $w \in L(\mathcal{P})$, there is an accepting run $\rho = c_0\tau_0\cdots\tau_n c_n$ on w that has no trailing ε -transitions, i.e.

1. $n = 0$ if $w = \varepsilon$ (which implies that c_0 is accepting), and
2. $\ell(\tau_0\cdots\tau_{n-1})$ is a strict prefix of w , if $w \neq \varepsilon$,
and $\tau_{n'} = r(\tau_0\cdots\tau_{n'-1}, w(|\ell(\tau_0\cdots\tau_{n'-1})|))$ for all $0 \leq n' < n$. If w is nonempty, then $w(|\ell(\tau_0\cdots\tau_{n'-1})|)$ is defined for all $0 \leq n' < n$ by the second requirement. Note that ρ is unique if it exists.

Note that the prefix processed so far can be recovered from r 's input, i.e. it is $\ell(\rho)$. However, the converse is not true due to the existence of ε -transitions. This is the reason that the run prefix and not the input prefix is the argument for the resolver. We denote the class of languages recognised by GFG-PDA by GFG-CFL.

Intuitively, every DPDA *should* be good-for-games, as there is no nondeterminism to resolve during a run. However, in order to reach a final state, a run of a DPDA on some input w may traverse *trailing* ε -transitions after the last letter of w is processed. On the other hand, the run of a GFG-PDA on w consistent with any resolver has to end with the transition processing the last letter of w . Hence, not every DPDA recognises the same language when viewed as a GFG-PDA. Nevertheless, we show, using standard pushdown automata constructions, that every DPDA can be turned into an equivalent GFG-PDA. As every GFG-PDA is a PDA by definition, we obtain a hierarchy of languages.

► **Lemma 3.** $DCFL \subseteq GFG-CFL \subseteq CFL$.

Instead of requiring that GFG-PDA end their run with the last letter processed, one could add an end-of-word marker that allows traversing trailing ε -transitions after the last letter has been processed. In Appendix A.1, we show that this alternative definition does not increase expressiveness, which explains our (arguably simpler) definition.

Finally, let us remark that GFGness of PDA and context-free languages is undecidable. These problems were shown to be undecidable for ω -GFG-PDA and ω -GFG-CFL by reductions from the inclusion and universality problem for PDA on finite words [25]. The same reductions also show that these problems are undecidable over PDA on finite words.

► **Theorem 4.** *The following problems are undecidable:*

1. Given a PDA \mathcal{P} , is \mathcal{P} a GFG-PDA?
2. Given a PDA \mathcal{P} , is $L(\mathcal{P}) \in GFG-CFL$?

2.3 Games and Universality

One of the motivations for GFG automata is that solving games with winning conditions given by a GFG automaton is easier than for nondeterministic automata. This makes them appealing for applications such as the synthesis of reactive systems, which can be modelled as a game between an antagonistic environment and the system. Solving games is undecidable for PDA in general [11], both over finite and infinite words, while for ω -GFG-PDA, it is EXPTIME-complete [25]. As a corollary, universality is also decidable for ω -GFG-PDA, while it is undecidable for PDA, both over finite and infinite words [19].

Here, we consider Gale-Stewart games [12], abstract games induced by a language in which two players alternately pick letters, thereby constructing an infinite word. One player aims to construct a word that is in the language while the other aims to construct one that is not in the language. Note that these games are different, but related, to games played on configuration graphs of pushdown automata [39].

Formally, given a language $L \subseteq (\Sigma_1 \times \Sigma_2)^*$ of sequences of letter pairs, the game $G(L)$ is played between Player 1 and Player 2 in rounds $i = 0, 1, \dots$ as follows: At each round i , Player 1 plays a letter $a_i \in \Sigma_1$ and Player 2 answers with a letter $b_i \in \Sigma_2$. A play of $G(L)$ is an infinite word $\binom{a_0}{b_0} \binom{a_1}{b_1} \dots$ and Player 2 wins such a play if and only if each of its prefixes is in the language L . A strategy for Player 2 is a mapping from Σ_1^+ to Σ_2 that gives for each prefix played by Player 1 the next letter to play. A play agrees with a strategy σ if for each i , $b_i = \sigma(a_0 a_1 \dots a_i)$. Player 2 wins $G(L)$ if she has a strategy that only agrees with plays that are winning for Player 2. Observe that Player 2 loses whenever the projection of L onto its first component is not universal. Finally, universality reduces to solving these games: \mathcal{P} is universal if and only if Player 2 wins $G(L)$ for $L = \{ \binom{w(0)}{\#} \dots \binom{w(n)}{\#} \mid w(0) \dots w(n) \in L(\mathcal{P}) \}$.

We now argue that solving games for GFG-PDA easily reduces to the case of ω -GFG-PDA, which are just GFG-PDA over infinite words, where acceptance is not determined by final state, since runs are infinite, but rather by the states or transitions visited infinitely often.

Here, we only need safety ω -GFG-PDA, in which every infinite run is accepting (i.e. rejection is implemented via missing transitions). The infinite Gale-Stewart game over a language L of infinite words, also denoted by $G(L)$, is as above, except that victory is determined by whether the infinite word built along the play is in L .

► **Lemma 5.** *Given a GFG-PDA \mathcal{P} , there is a safety ω -GFG-PDA \mathcal{P}' no larger than \mathcal{P} such that Player 2 wins $G(L(\mathcal{P}))$ if and only if she wins $G(L(\mathcal{P}'))$.*

Proof. Let \mathcal{P}' be the PDA obtained from \mathcal{P} by removing all transitions (q, X, a, q', γ) of \mathcal{P} with $a \in \Sigma$ and with non-final q' .

With a safety condition, in which every infinite run is accepting, \mathcal{P}' recognises exactly those infinite words whose prefixes are all accepted by \mathcal{P} . Hence, the games $G(L(\mathcal{P}))$ and $G(L(\mathcal{P}'))$ have the same winning player. Note that the correctness of this construction crucially relies on our definition of GFG-PDA, which requires a run on a finite word to end as soon as the last letter is processed. Then, the word is accepted if and only if the state reached by processing this last letter is final.

Finally, since \mathcal{P} is GFG, so is \mathcal{P}' . Consider an infinite input in $L(\mathcal{P}')$. Then, every prefix w has an accepting run of \mathcal{P} induced by its resolver, which implies that the last transition of this run (which processes the last letter of w) is not one of those that are removed to obtain \mathcal{P}' . Now, an induction shows that the same resolver works for \mathcal{P}' as well, relying on the fact that if w and w' with $|w| < |w'|$ are two such prefixes, then the resolver-induced run of \mathcal{P} on w is a prefix of the resolver-induced run of \mathcal{P} on w' . ◀

Our main results of this section are now direct consequences of the corresponding results on ω -words [25].

► **Corollary 6.** *Given a GFG-PDA \mathcal{P} , deciding whether $L(\mathcal{P}) = \Sigma^*$ and whether Player 2 wins $G(L(\mathcal{P}))$ are both in EXPTIME.*

2.4 Closure properties

Like ω -GFG-PDA, GFG-PDA have poor closure properties.

► **Theorem 7.** *GFG-PDA are not closed under union, intersection, complementation, set difference and homomorphism.*

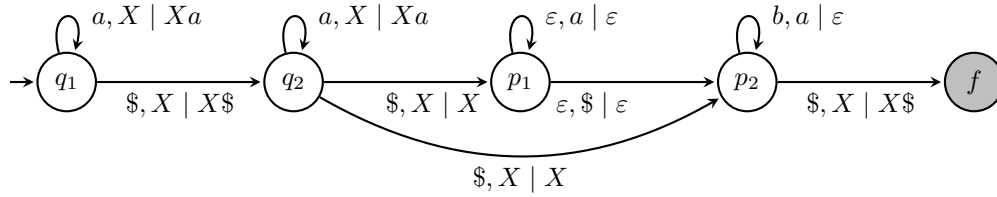
The proofs are similar to those used for ω -GFG-PDA and relegated to the full version [14]. There, we also study the closure properties under these operations with regular languages: If L is in GFG-CFL and R is regular, then $L \cup R$, $L \cap R$ and $L \setminus R$ are also in GFG-CFL, but $R \setminus L$ is not necessarily in GFG-CFL.

3 Expressiveness

Here we show that GFG-PDA are more expressive than DPDA but less expressive than PDA.

► **Theorem 8.** $DCFL \subsetneq GFG-CFL \subsetneq CFL$.

To show that GFG-PDA are more expressive than deterministic ones, we consider the language $B_2 = \{a^i \$ a^j \$ b^k \$ \mid k \leq \max(i, j)\}$. It is recognised by the PDA \mathcal{P}_{B_2} depicted in Figure 2, hence $B_2 \in CFL$. The first two states q_1 and q_2 deterministically push the input onto the stack, until the occurrence of the second $\$$. When the second $\$$ is processed, there is a nondeterministic choice to move to p_1 or p_2 and erase along ε -transitions 1 or 0 blocks



■ **Figure 2** A PDA \mathcal{P}_{B_2} recognising B_2 . Grey states are final, and X is an arbitrary stack symbol.

from the stack, so that the 1st or 2nd block of a 's respectively remains at the top of the stack. Then, the automaton compares the length of the b -block in the input with the length of the a -block at the top of the stack and accepts if the b -block is shorter, i.e. the third $\$$ is processed before the whole a -block is popped off the stack. If the input has not the form $a^i\$a^j\$b^k\$\$, then it is rejected.

We show that $B_2 \in \text{GFG-CFL}$ by proving that \mathcal{P}_{B_2} is good-for-games: the nondeterministic choice between moving to p_1 or to p_2 can be made only based on the prefix $a^i\$a^j$ processed so far. This is straightforward, as a resolver only needs to know which of i and j is larger, which can be determined from the run prefix constructed thus far. Then, in order to show that B_2 is not in DCFL , we prove that its complement B_2^c is not a context-free language. Since DCFL is closed under complementation, this implies the desired result.

Finally, to show that PDA are more expressive than GFG-PDA, we consider the language $L = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$. We note that $L \in \text{CFL}$ while we show below $L \notin \text{GFG-CFL}$.

Unambiguous context-free languages, i.e. those generated by grammars for which every word in the language has a unique leftmost derivation, are another class sitting between DCFL and CFL . Thus, it is natural to ask how unambiguity and GFGness are related: To conclude this section, we show that both notions are independent.

► **Theorem 9.** *There is an unambiguous context-free language that is not in GFG-CFL and a language in GFG-CFL that is inherently ambiguous.*

An unambiguous grammar for the language $\{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\} \notin \text{GFG-CFL}$ is easy to construct and we show that the language $B = \{a^i b^j c^k \mid i, j, k \geq 1, k \leq \max(i, j)\}$ is inherently ambiguous. Its inclusion in GFG-CFL is easily established using a similar argument as for the language $B_2 = \{a^i\$a^j\$b^k\$\mid k \leq \max(i, j)\}$ above. The dollars add clarity to the GFG-PDA but are cumbersome in the proof of inherent ambiguity.

4 Succinctness

We show that GFG-PDA are not only more expressive than DPDA, but also more succinct. Similarly, we show that PDA are more succinct than GFG-PDA.

► **Theorem 10.** *GFG-PDA can be exponentially more succinct than DPDA, and PDA can be double-exponentially more succinct than GFG-PDA.*

We first show that GFG-PDA can be exponentially more succinct than DPDA. To this end, we construct a family $(C_n)_{n \in \mathbb{N}}$ of languages such that C_n is recognised by a GFG-DPDA of size $O(n)$, yet every DPDA recognising C_n has at least exponential size in n .

Let $c_n \in (\{0, 1\}^n)^*$ be the word describing an n -bit binary counter counting from 0 to $2^n - 1$. For example, $c_2 = \$00\$01\$10\11 . We consider the family of languages $C_n = \{w \in \{0, 1, \$, \#\}^* \mid w \neq c_n\#\} \subseteq \{0, 1, \$, \#\}^*$ of bad counters.

We show that the language C_n is recognised by a GFG-PDA of size $O(n)$ and that every DPDA \mathcal{D} recognising C_n has exponential size in n . Observe that this result implies that even GFG-PDA that are equivalent to DPDA are not determinisable by pruning. In contrast, for NFA, GFGness implies determinisability by pruning [6].

We conclude this section by showing that PDA can be double-exponentially more succinct than GFG-PDA. We show that there exists a family $(L_n)_{n>0}$ of languages such that L_n is recognised by a PDA of size $O(\log n)$ while every GFG-PDA recognising this language has at least exponential size in n .

Formally, we set $L_n = (0+1)^*1(0+1)^{n-1}$, that is, the n^{th} bit from the end is a 1. We count starting from 1, so that the last bit is the 1st bit from the end. Note that this is the standard example for showing that NFA can be exponentially more succinct than DFA, and has been used for many other succinctness results ever since.

5 Good-for-games Visibly Pushdown Automata

One downside of GFG-PDA is that, like ω -GFG-PDA, they have poor closure properties and checking GFGness is undecidable. We therefore consider a well-behaved class of GFG-PDA, namely GFG visibly pushdown automata, GFG-VPA for short, that is closed under union, intersection, and complementation.

Let Σ_c, Σ_r and Σ_{int} be three disjoint sets of *call* symbols, *return* symbols and *internal* symbols respectively. Let $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_{\text{int}}$. A *visibly pushdown automaton* [2] (VPA) $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ is a restricted PDA that pushes onto the stack only when it reads a call symbol, it pops the stack only when a return symbol is read, and does not use the stack when there is an internal symbol. Formally,

- a letter $a \in \Sigma_c$ is only processed by transitions of the form (q, X, a, q', XY) with $X \in \Gamma_{\perp}$, i.e. some stack symbol $Y \in \Gamma$ is pushed onto the stack.
- A letter $a \in \Sigma_r$ is only processed by transitions of the form $(q, X, a, q', \varepsilon)$ with $X \neq \perp$ or (q, \perp, a, q', \perp) , i.e. the topmost stack symbol is removed, or if the stack is empty, it is left unchanged.
- A letter $a \in \Sigma_{\text{int}}$ is only processed by transitions of the form (q, X, a, q', X) with $X \in \Gamma_{\perp}$, i.e. the stack is left unchanged.
- There are no ε -transitions.

Intuitively, the stack height of the last configuration of a run processing some $w \in (\Sigma_c \cup \Sigma_r \cup \Sigma_s)^*$ only depends on w .

We denote by GFG-VPA the VPA that are good-for-games. Every VPA (and hence every GFG-VPA) can be determinised, i.e. all three classes of automata recognise the same class of languages, denoted by VPL, which is a strict subset of DCFL [2]. However, VPA can be exponentially more succinct than deterministic VPA (DVPA) [2]. We show that there is an exponential gap both between the succinctness of GFG-VPA and DVPA and between VPA and GFG-VPA. The proof of the former gap again uses a language of bad counters, similar to C_n used in Theorem 10, which we adapt for the VPA setting by adding a suffix allowing the automaton to pop the stack. Furthermore, for the gap between VPA and GFG-VPA, we similarly adapt the language L_n of words where the n^{th} bit from the end is a 1, from the proof of Theorem 10, by making sure that the stack height is always bounded by 1. Then, a GFG-VPA is essentially a GFG-NFA, and therefore determinisable by pruning, which means that it is as big as a deterministic automaton for the language.

► **Theorem 11.** *GFG-VPA can be exponentially more succinct than DVPA and VPA can be exponentially more succinct than GFG-VPA.*

We now turn to the question of deciding whether a given VPA is GFG. We show decidability using the *one-token game*, introduced by Bagnol and Kuperberg [3]. It modifies the game-based characterisation of GFGness of ω -regular automata by Henzinger and Piterman [16]. While the one-token game does not characterise the GFGness of Büchi automata, here we show that it suffices for VPA. The matching lower bound follows from a reduction from the inclusion problem for VPA, which is EXPTIME-hard [2], to GFGness (see [25] for details of the reduction in the context of ω -GFG-PDA).

► **Theorem 12.** *The following problem is EXPTIME-complete: Given a VPA \mathcal{P} , is \mathcal{P} GFG?*

We first define the *one-token game*, introduced by Bagnol and Kuperberg [3] in the context of regular languages, for VPA. Given a VPA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$, the positions of the one-token game consist of pairs of configurations (c_i, c'_i) , starting from the initial configuration of \mathcal{P} . At each round i :

- Player 1 picks a letter $a_i \in \Sigma$,
- Player 2 picks an a_i -transition $\tau_i \in \Delta$ enabled in c_i , leading to a configuration c_{i+1} ,
- Player 1 picks an a_i -transition $\tau'_i \in \Delta$ enabled in c'_i , leading to a configuration c'_{i+1} ,
- The game proceeds from the configuration (c_{i+1}, c'_{i+1}) .

A play consists of an infinite word $a_0 a_1 \dots \in \Sigma^\omega$ and two sequences of transitions $\tau_0 \tau_1 \dots$ and $\tau'_0 \tau'_1 \dots$ built by Players 2 and 1 respectively. Player 1 wins if for some n , $\tau'_0 \dots \tau'_n$ is an accepting run of \mathcal{P} over $a_0 \dots a_n$ and $\tau_0 \dots \tau_n$ is not. Recall that VPA do not have ε -transitions, so the two runs proceed in lockstep.

Observe that this game can be seen as a safety game on a visibly pushdown arena and can therefore be encoded as a Gale-Stewart game with a DCFL winning condition. This in turn is solvable in EXPTIME [39]. To prove Theorem 12, it now suffices to argue that this game characterises whether the VPA \mathcal{P} is GFG.

Proof. We now argue that \mathcal{P} is GFG if and only if Player 2 wins the one-token game on \mathcal{P} . One direction is immediate: if \mathcal{P} is GFG, then the resolver is also a strategy for Player 2 in the one-token game.

For the converse direction, consider the family of *copycat strategies* for Player 1 that copy the transition chosen by Player 2 until she plays an a -transition from a configuration c to a configuration c' such that there is a word aw that is accepted from c but w is not accepted from c' . We call such transitions non-residual. If Player 2 plays such a non-residual transition, then the copycat strategies stop copying and instead play the letters of w and the transitions of an accepting run over aw from c .

If Player 2 wins the one-token game with a strategy s , she wins, in particular, against this family of copycat strategies for Player 1. Observe that copycat strategies win any play along which Player 2 plays a non-residual transition. Therefore s must avoid ever playing a non-residual transition. We can now use s to induce a resolver r_s for \mathcal{P} : r_s maps a sequence of transitions over a word w to the transition chosen by s in the one-token game where Player 1 played w and a copycat strategy. Then, r_s never produces a non-residual transition. As a result, if a word w is in $L(\mathcal{P})$, then the run induced by r_s over every prefix v of w leads to a configuration that accepts the remainder of w . This is in particular the case for w itself, for which r_s induces an accepting run. This concludes our argument that r_s is indeed a resolver, and \mathcal{P} is therefore GFG.

Thus, to decide whether a VPA \mathcal{P} is GFG it suffices to solve the one-token game on \mathcal{P} , which can be done in exponential time. ◀

Finally, we relate the GFGness problem to the *good-enough synthesis* problem [1], also known as the *uniformization* problem [9], which is similar to the Church synthesis problem, except that the system is only required to satisfy the specification on inputs in the projection of the specification on the first component.

Let $w \in \Sigma_1$ and $w' \in \Sigma_2$ with $|w| = |w'|$. Then, for the sake of readability, we write $\binom{w}{w'}$ for the word $\binom{w(0)}{w'(0)} \cdots \binom{w(|w|-1)}{w'(|w|-1)}$ over $\Sigma_1 \times \Sigma_2$.

► **Definition 13** (GE-synthesis). *Given a language $L \subseteq (\Sigma_1 \times \Sigma_2)^*$, is there a function $f : \Sigma_1^* \rightarrow \Sigma_2$ such that for each $w \in \{w \mid \exists w' \in \Sigma_2^*. \binom{w}{w'} \in L\}$ the word $\binom{w}{w'}$ is in L , where $w'(n) = f(w(0) \cdots w(n))$ for each $0 \leq n < |w|$.*

We now prove that the GE-synthesis problem for GFG-VPA and DVPA is as hard as the GFGness problem for VPA, giving us the following corollary of Theorem 12.

► **Corollary 14.** *The GE-synthesis problem for inputs given by GFG-VPA, and in particular for DVPA, is EXPTIME-complete.*

Proof. We first reduce the good-enough synthesis problem to the GFGness problem. Given a GFG-VPA $\mathcal{P} = (Q, \Sigma_1 \times \Sigma_2, \Gamma, q_I, \Delta, F)$, with resolver r , let \mathcal{P}' be \mathcal{P} projected onto the first component: $\mathcal{P}' = (Q, \Sigma_1, \Gamma, q_I, \Delta', F)$ has the same states, stack alphabet and final states as \mathcal{P} , but has an a -transition for some $a \in \Sigma_1$ whenever \mathcal{P} has the same transition over $\binom{a}{b}$ for some $b \in \Sigma_2$. Let each transition of \mathcal{P}' be annotated with the Σ_2 -letter of the corresponding \mathcal{P} -transition. Thus \mathcal{P}' recognises the projection of $L(\mathcal{P})$ on the first component.

A resolver for \mathcal{P}' induces a GE-synthesis function for \mathcal{P} by reading the Σ_2 -annotation of the chosen transitions in \mathcal{P}' . Indeed, the resolver produces an accepting run with annotation w' of \mathcal{P}' for every word w in the projection of $L(\mathcal{P})$ on the first component. The same run is an accepting run in \mathcal{P} over $\binom{w}{w'}$ which is therefore in $L(\mathcal{P})$. Conversely a GE-synthesis function f for \mathcal{P} , combined with r , induces a resolver r' for \mathcal{P}' by using f to choose output letters and r to choose which transition of \mathcal{P} to use; together these uniquely determine a transition in \mathcal{P}' . Then, if $w \in L(\mathcal{P}')$, f guarantees that the annotation of the run induced by r' in \mathcal{P}' is a witness w' such that $\binom{w}{w'} \in \mathcal{P}$, and then r guarantees that the run is accepting, since the corresponding run in \mathcal{P} over $\binom{w}{w'}$ must be accepting.

We now reduce the GFGness problem of a VPA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ to the GE-synthesis problem of a DVPA $\mathcal{P}' = (Q, \Sigma \times \Delta, \Gamma, q_I, \Delta', F)$. The deterministic automaton \mathcal{P}' is as \mathcal{P} except that each transition τ over a letter a in Δ is replaced with the same transition over $\binom{a}{\tau}$ in Δ' . In other words, \mathcal{P}' recognises the accepting runs of \mathcal{P} and its GE-synthesis problem asks whether there is a function that constructs on-the-fly an accepting run for every word in $L(\mathcal{P})$, that is, whether \mathcal{P} has a resolver. ◀

In contrast, for LTL specifications, the GE-synthesis problem is 2EXPTIME-complete [1].

6 Resolvers

The definition of a resolver does not put any restrictions on its complexity. In this section we study the complexity of the resolvers that GFG-PDA need. We consider two somewhat orthogonal notions of complexity: memory and machinery. On one hand, we show that resolvers can always be chosen to be *positional*, that is, dependent on the current state

and stack configuration only. Note that this is not the case for ω -regular automata², let alone ω -GFG-PDA. On the other hand, we show that they are not always implementable by pushdown transducers.

More formally, a resolver r is positional, if for any two sequences ρ and ρ' of transitions inducing runs ending in the same configuration, $r(\rho, a) = r(\rho', a)$ for all $a \in \Sigma$.

► **Lemma 15.** *Every GFG-PDA has a positional resolver.*

Proof. Let r' be a (not necessarily positional) resolver for \mathcal{P} . We define a resolver r such that for each configuration and input letter, it makes a choice consistent with r' for some input leading to this configuration. In other words, for every reachable configuration c , let ρ_c be an input to r' inducing a run ending in c . Then, we define $r(\rho, a) = r(\rho_c, a)$, where c is the last configuration of the run induced by ρ .

We claim that r , which is positional by definition, is a resolver. Towards a contradiction, assume that this is not the case, i.e. there is a word $w \in L(\mathcal{P})$ such that the run ρ induced by r is rejecting. Since this run is finite and $w \in L(\mathcal{P})$, there is some last configuration c along the run ρ from which the rest of the word, say u , is accepted³ (by some other run of \mathcal{P} having the same prefix as ρ up to configuration c). Let τ be the next transition along ρ from c . Since r chose τ , the resolver r' also chooses τ after some history leading to c , over some word v . Since u is accepted from c , the word vu is in $L(\mathcal{P})$; since r' is a resolver, there is an accepting run over u from c starting with τ , contradicting that c is the last position on ρ from where the rest of the word could be accepted. ◀

Contrary to the case of finite and ω -regular automata, since GFG-PDA have an infinite configuration space, the existence of positional resolvers does not imply determinisability. On the other hand, if a GFG-PDA has a resolver which only depends on the mode of the current configuration, then it is *determinisable by pruning*, as transitions that are not used by the resolver can be removed to obtain a deterministic automaton. However, not all GFG-PDA are determinisable by pruning, e.g. the GFG-PDA for the languages C_n used to prove Theorem 10.

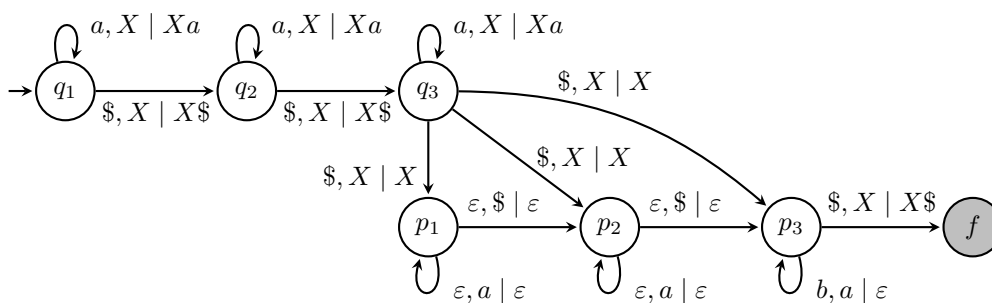
We now turn to how powerful resolvers for GFG-PDA need to be. First, we introduce transducers as a way to implement a resolver. A transducer is an automaton with outputs instead of acceptance, i.e., it computes a function from input sequences to outputs. A pushdown resolver is a pushdown transducer that implements a resolver.

Note that a resolver has to pick enabled transitions in order to induce accepting runs for all inputs in the language. To do so, it needs access to the mode of the last configuration. However, to keep track of this information on its own, the pushdown resolver would need to simulate the stack of the GFG-PDA it controls. This severely limits the ability of the pushdown resolver to implement computations on its own stack. Thus, we give a pushdown resolver access to the current mode of the GFG-PDA via its output function, thereby freeing its own stack to implement further functionalities.

Formally, a pushdown transducer (PDT for short) $\mathcal{T} = (\mathcal{D}, \lambda)$ consists of a DPDA \mathcal{D} augmented with an output function $\lambda : Q^{\mathcal{D}} \rightarrow \Theta$ mapping the states $Q^{\mathcal{D}}$ of \mathcal{D} to an output alphabet Θ . The input alphabet of \mathcal{T} is the input alphabet of \mathcal{D} .

² A positional resolver for ω -regular automata implies determinisability by pruning, and we know that this is not always possible [6].

³ Observe that this is no longer true over infinite words as an infinite run can stay within configurations from where an accepting run exists without being itself accepting. In fact, the lemma does not even hold for coBüchi automata [23] as the existence of positional resolvers implies determinisability by pruning.



■ **Figure 3** The PDA \mathcal{P}_{B_3} for B_3 . Grey states are final, and X is an arbitrary stack symbol.

Then, given a PDA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$, a pushdown resolver for \mathcal{P} consists of a pushdown transducer $\mathcal{T} = (\mathcal{D}, \lambda)$ with input alphabet Δ and output alphabet $Q \times \Gamma_{\perp} \times \Sigma \rightarrow \Delta$ such that the function $r_{\mathcal{T}}$, defined as follows, is a resolver for \mathcal{P} : $r_{\mathcal{T}}(\tau_0 \dots \tau_k, a) = \lambda(q_{\mathcal{T}})(q_{\mathcal{P}}, X, a)$ where

- $q_{\mathcal{T}}$ is the state of the last configuration of the longest run of \mathcal{D} on $\tau_0 \dots \tau_k$ (recall that while \mathcal{D} is deterministic, it may have several runs on an input which differ on trailing ϵ -transitions);
- $(q_{\mathcal{P}}, X)$ is the mode of the last configuration of the run of \mathcal{P} induced by $\tau_0 \dots \tau_k$.

In other words, a transducer implements a resolver by processing the run so far, and then uses the output of the state reached and the state and top stack symbol of the GFG-PDA to determine the next transition in the GFG-PDA.

We now give an example of a GFG-PDA which does not have a pushdown resolver. The language in question is the language $B_3 = \{a^i \$ a^j \$ a^k \$ b^l \$ \mid l \leq \max(i, j, k)\}$. Compare this to the language B_2 in Section 3 which *does* have a pushdown resolver. Let \mathcal{P}_{B_3} be the automaton in Figure 3, which works analogously to the automaton for B_2 in Figure 2.

This automaton recognises B_3 : for a run to end in the final state, the stack, and therefore the input, must have had an a -block longer than or equal to the final b -block; conversely, if the b -block is shorter than or equal to some a -block, the automaton can nondeterministically pop the blocks on top of the longest a -block off the stack before processing the b -block. Furthermore, this automaton is GFG: the nondeterminism can be resolved by removing from the stack all blocks until the *longest* a -block is at the top of the stack, and this choice can be made once the third $\$$ is processed.

We now argue that this GFG-PDA needs more than a pushdown resolver. The reason is that a pushdown resolver needs to be able to determine which of the three blocks is the longest while processing a prefix of the form $a^* \$ a^* \$ a^*$. However, at least one of the languages induced by these three choices is not context-free, yielding the desired contradiction.

► **Lemma 16.** *The GFG-PDA \mathcal{P}_{B_3} has no pushdown resolver.*

Proof. Towards a contradiction, assume that there is a pushdown resolver r for \mathcal{P}_{B_3} , implemented by a PDT $\mathcal{T} = (\mathcal{D}, \lambda)$.

From \mathcal{T} , for each $i \in \{1, 2, 3\}$, we can construct a PDA \mathcal{D}_i that recognises the language of words $w \in a^* \$ a^* \$ a^*$ such that \mathcal{T} chooses from q_3 the transition of \mathcal{P}_{B_3} going to p_i when constructing a run on $w \$$: this is simply the pushdown automaton \mathcal{D} underlying \mathcal{T} where transitions of \mathcal{D} processing non- ϵ transitions τ of \mathcal{P}_{B_3} are modified to now process $\ell(\tau) \in \{a, b, \$\}$, transitions of \mathcal{D} processing ϵ -transitions of \mathcal{P}_{B_3} are removed, and states q of \mathcal{D} such that $\lambda(q)(q_3, X, \$) = (q_3, \$, X, p_i, X)$ are made final, intersected with a DFA checking that the input is in $a^* \$ a^* \$ a^*$.

Since \mathcal{T} implements a resolver for \mathcal{P} , each \mathcal{D}_i only accepts words of the form $a^{m_1}\$a^{m_2}\a^{m_3} such that $\max(m_1, m_2, m_3) = m_i$. Furthermore, at least for one $i \in \{1, 2, 3\}$, \mathcal{D}_i accepts $a^m\$a^m\a^m for infinitely many m .

To reach a contradiction, we now argue that this \mathcal{D}_i recognises a language that is not context-free. Indeed, if it were, then by applying the pumping lemma for context-free languages, there would be a large enough m such that the word $a^m\$a^m\$a^m \in L(\mathcal{D}_i)$ could be decomposed as $uvwyz$ such that $|vy| \geq 1$ and $uv^nwy^n z$ is in the language of \mathcal{D}_i for all $n \geq 0$. In this decomposition, v and y must be $\$$ -free. Then, if either v or y occurs in the i^{th} block and is non-empty, by setting $n = 0$ we obtain a contradiction as the i^{th} block is no longer the longest. Otherwise, we obtain a similar contradiction by setting $n = 2$. In either case, this shows that \mathcal{T} is not a pushdown resolver for \mathcal{P} . ◀

Another restricted class of resolvers are finite-state resolvers, which can be seen as pushdown resolvers that do not use their stack. Similarly to the case of ω -GFG-PDA [26], the product of a GFG-PDA and a finite-state resolver yields a DPDA for the same language.

► **Remark 17.** Every GFG-PDA with a finite-state resolver is determinisable.

Note that the converse does not hold. For example, consider the regular, and therefore deterministic context-free, language L_{10} of words $w\#$ with $w \in \{a, b\}^*$ with infix a^{10} . A GFG-PDA \mathcal{P}_{10} recognising L_{10} can be constructed as follows: \mathcal{P}_{10} pushes its input onto its stack until processing the first $\#$. Before processing this letter, \mathcal{P}_{10} uses ε -transitions to empty the stack again. While doing so, it can nondeterministically guess whether the next 10 letters removed from the stack are all a 's. If yes, it accepts; in all other cases (in particular if the input word does not end with the first $\#$ or the infix a_{10} is not encountered on the stack) it rejects. This automaton is good-for-games, as a resolver has access to the whole prefix before the first $\#$ when searching for a^{10} while emptying the stack. This is sufficient to resolve the nondeterminism. On the other hand, there is no finite-state resolver for \mathcal{P}_{10} , as resolving the nondeterminism, intuitively, requires to keep track of the whole prefix before the first $\#$ (recall that a finite-state resolver only has access to the topmost stack symbol).

In Appendix A.2 we consider another model of pushdown resolver, namely one that does not only have access to the mode of the GFG-PDA, but can check the full stack for regular properties. We show that this change does not increase the class of good-for-games context-free languages that are recognised by a GFG-PDA with a pushdown resolver.

Finally, for GFG-VPA, the situation is again much better. The classical game-based characterisation of GFGness of ω -regular automata by Henzinger and Piterman [16] can be lifted to VPA. Then, using known results [27] about VPA games having VPA strategies, we obtain our final theorem.

► **Theorem 18.** *Every GFG-VPA has a (visibly) pushdown resolver.*

Proof. Fix a VPA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ and consider the following two-player game $\mathcal{G}(\mathcal{P})$, introduced by Henzinger and Piterman to decide GFGness of ω -automata [16]. In each round, first Player 1 picks a letter from Σ or ends the play. If he has not ended the play, then Player 2 picks a transition of \mathcal{P} . Hence, once Player 1 has stopped the play, Player 1 has picked an input word w over Σ^* and Player 2 has indicated a run ρ of \mathcal{P} . A finite play with outcome (w, ρ) is winning for Player 2 if either $w \notin L(\mathcal{P})$ or ρ induces an accepting run of \mathcal{P} on w . A strategy for Player 2 in this game is a mapping $\sigma: \Sigma^+ \rightarrow \Delta$ and an outcome $(w(0) \cdots w(k), \rho(0) \cdots \rho(k))$ is consistent with σ , if $\rho(j) = \sigma(w(0) \cdots w(j))$ for every $0 \leq j \leq k$. We say that σ is winning for Player 2, if every outcome of a finite play that is consistent with σ is winning for her (note that we disregard infinite plays).

Now, Player 2 wins $\mathcal{G}(\mathcal{P})$ if and only if \mathcal{P} is a GFG-VPA. This follows as every winning strategy for Player 2 can be turned into a resolver and vice versa.

Now, as the class of languages recognized by VPA, is closed under complementation and union [2], one can encode $\mathcal{G}(\mathcal{P})$ as a Gale-Stewart game with a VPL winning condition. Such games can be solved effectively [27] and the winner always has a winning strategy implemented by a (visibly) PDT. Thus, if \mathcal{P} is a GFG-VPA, i.e. Player 2 wins $\mathcal{G}(\mathcal{P})$, then she has a winning strategy implemented by a (visibly) PDT, which can easily be turned into a (visibly) pushdown resolver for \mathcal{P} . ◀

7 Conclusion

We have continued the study of good-for-games pushdown automata, focusing on expressiveness and succinctness. In particular, we have shown that GFG-PDA are not only more expressive than DPDA (as had already been shown for the case of infinite words), but also more succinct than DPDA: We have introduced the first techniques for using GFG nondeterminism to succinctly represent languages that do not depend on the coBüchi condition. Similarly, for the case of VPA, for which deterministic and nondeterministic automata are equally expressive, we proved a (tight) exponential gap in succinctness.

Solving games and universality are decidable for GFG-PDA, but GFGness is undecidable and GFG-PDA have limited closure properties. On the other hand, GFGness for VPA is decidable and they inherit the closure properties of VPA, e.g. union, intersection and complementation, making GFG-VPA an exciting class of pushdown automata. Finally, we have studied the complexity of resolvers for GFG-PDA, showing that positional ones always suffice, but that they are not always implementable by pushdown transducers. Again, GFG-VPA are better-behaved, as they always have a resolver implementable by a VPA.

Let us conclude by mentioning some open problems raised by our work.

- It is known that the succinctness gap between PDA and DPDA is noncomputable [15, 38], i.e. there is no computable function f such that any PDA of size n that has some equivalent DPDA also has an equivalent DPDA of size $f(n)$. Due to our hierarchy results, at least one of the succinctness gaps between PDA and GFG-PDA and between GFG-PDA and DPDA has to be uncomputable, possibly both.
- We have shown that some GFG-PDA do not have pushdown resolvers. It is even open whether every GFG-PDA has a computable resolver.
- On the level of languages, it is open whether every language in GFG-CFL has a GFG-PDA recognising it with a resolver implementable by a pushdown transducer.
- We have shown that GFGness is undecidable, both for PDA and for context-free languages. Is it decidable whether a given GFG-PDA has an equivalent DPDA?
- Equivalence of DPDA is famously decidable [36] while it is undecidable for PDA [19]. Is equivalence of GFG-PDA decidable?
- Does every GFG-PDA that is equivalent to a DPDA have a finite-state resolver with regular stack access (see Appendix A.2 for definitions)?
- There is a plethora of fragments of context-free languages one can compare GFG-CFL to, let us just mention a few interesting ones: Height-deterministic context-free languages [30], context-free languages with bounded nondeterminism [17] and preorder typeable visibly pushdown languages [21].

References

- 1 Shaull Almagor and Orna Kupferman. Good-enough synthesis. In Shuvendu K. Lahiri and Chao Wang, editors, *CAV 2020, Part II*, volume 12225 of *LNCS*, pages 541–563. Springer, 2020. doi:10.1007/978-3-030-53291-8_28.
- 2 Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In László Babai, editor, *STOC 2004*, pages 202–211. ACM, 2004. doi:10.1145/1007352.1007390.
- 3 Marc Bagnol and Denis Kuperberg. Büchi Good-for-Games Automata Are Efficiently Recognizable. In Sumit Ganguly and Paritosh Pandya, editors, *FSTTCS 2018*, volume 122 of *LIPICs*, pages 16:1–16:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.FSTTCS.2018.16.
- 4 Henrik Björklund and Wim Martens. The tractability frontier for NFA minimization. *Journal of Computer and System Sciences*, 78(1):198–210, 2012.
- 5 Henrik Björklund, Martin Schuster, Thomas Schwentick, and Joscha Kulbatzki. On optimum left-to-right strategies for active context-free games. In Wang-Chiew Tan, Giovanna Guerrini, Barbara Catania, and Anastasios Gounaris, editors, *ICDT 2013*, pages 105–116. ACM, 2013. doi:10.1145/2448496.2448510.
- 6 Udi Boker, Orna Kupferman, and Michał Skrzypczak. How deterministic are good-for-games automata? *arXiv*, 2017. arXiv:1710.04115.
- 7 J. Richard Büchi. Regular canonical systems. *Archiv für mathematische Logik und Grundlagenforschung*, 6(3):91–111, April 1964. doi:10.1007/BF01969548.
- 8 Arnaud Carayol and Matthew Hague. Saturation algorithms for model-checking pushdown systems. In Zoltán Ésik and Zoltán Fülöp, editors, *AFL 2014*, volume 151 of *EPTCS*, pages 1–24, 2014. doi:10.4204/EPTCS.151.1.
- 9 Arnaud Carayol and Christof Löding. Uniformization in automata theory, 2015. In *LMPS 2015*. URL: <https://hal.archives-ouvertes.fr/hal-01806575>.
- 10 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikolettseas, and Wolfgang Thomas, editors, *ICALP 2009, (Part II)*, volume 5556 of *LNCS*, pages 139–150. Springer, 2009. doi:10.1007/978-3-642-02930-1_12.
- 11 Olivier Finkel. Topological properties of omega context-free languages. *Theor. Comput. Sci.*, 262(1):669–697, 2001. doi:10.1016/S0304-3975(00)00405-9.
- 12 David Gale and F. M. Stewart. Infinite games with perfect information. In Harold William Kuhn and Albert William Tucker, editors, *Contributions to the Theory of Games (AM-28), Volume II*, chapter 13, pages 245–266. Princeton University Press, 1953.
- 13 Jonathan Goldstine, Hing Leung, and Detlef Wotschke. Measuring nondeterminism in pushdown automata. *Journal of Computer and System Sciences*, 71(4):440–466, 2005.
- 14 Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. A bit of nondeterminism makes pushdown automata expressive and succinct. *arXiv*, 2021. arXiv:2105.02611.
- 15 Juris Hartmanis. On the succinctness of different representations of languages. *SIAM J. Comput.*, 9(1):114–120, 1980. doi:10.1137/0209010.
- 16 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In Zoltán Ésik, editor, *CSL 2006*, volume 4207 of *LNCS*, pages 395–410. Springer, 2006. doi:10.1007/11874683_26.
- 17 Christian Herzog. Pushdown automata with bounded nondeterminism and bounded ambiguity. *Theor. Comput. Sci.*, 181(1):141–157, 1997.
- 18 Markus Holzer and Martin Kutrib. Descriptive complexity of (un)ambiguous finite state machines and pushdown automata. In Antonín Kucera and Igor Potapov, editors, *RP 2010*, volume 6227 of *LNCS*, pages 1–23. Springer, 2010. doi:10.1007/978-3-642-15349-5_1.
- 19 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

- 20 Tao Jiang and Bala Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.
- 21 Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Visibly pushdown automata for streaming XML. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *WWW 2007*, pages 1053–1062. ACM, 2007. doi:10.1145/1242572.1242714.
- 22 Denis Kuperberg and Anirban Majumdar. Computing the width of non-deterministic automata. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:10)2019.
- 23 Denis Kuperberg and Michal Skrzypczak. On determinisation of good-for-games automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015 (Part II)*, volume 9135 of *LNCS*, pages 299–310. Springer, 2015. doi:10.1007/978-3-662-47666-6_24.
- 24 Orna Kupferman, Shmuel Safra, and Moshe Y Vardi. Relating word and tree automata. *Annals of Pure and Applied Logic*, 138(1-3):126–146, 2006.
- 25 Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω -pushdown automata. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS 2020*, pages 689–702. ACM, 2020. doi:10.1145/3373718.3394737.
- 26 Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω -pushdown automata. *arXiv*, 2020. arXiv:2001.04392.
- 27 Christof Löding, P. Madhusudan, and Olivier Serre. Visibly pushdown games. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS 2004*, volume 3328 of *LNCS*, pages 408–420. Springer, 2004. doi:10.1007/978-3-540-30538-5_34.
- 28 Wim Martens, Frank Neven, Thomas Schwentick, and Geert Jan Bex. Expressiveness and complexity of XML schema. *TODS*, 31(3):770–813, 2006.
- 29 Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Active context-free games. *Theory of Computing Systems*, 39(1):237–276, 2006.
- 30 Dirk Nowotka and Jirí Srba. Height-deterministic pushdown automata. In Ludek Kucera and Antonín Kucera, editors, *MFCS 2007*, volume 4708 of *LNCS*, pages 125–134. Springer, 2007. doi:10.1007/978-3-540-74456-6_13.
- 31 Alexander Okhotin and Kai Salomaa. Descriptive complexity of unambiguous input-driven pushdown automata. *Theor. Comput. Sci.*, 566:1–11, 2015. doi:10.1016/j.tcs.2014.11.015.
- 32 Bader Abu Radi and Orna Kupferman. Minimizing GFG Transition-Based Automata. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *ICALP 2019*, volume 132 of *LIPICs*, pages 100:1–100:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ICALP.2019.100.
- 33 Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer, 1978. doi:10.1007/978-1-4612-6264-0.
- 34 Kai Salomaa and Sheng Yu. Limited nondeterminism for pushdown automata. *Bull. EATCS*, 50:186–193, 1993.
- 35 Sven Schewe. Minimising Good-For-Games Automata Is NP-Complete. In Nitin Saxena and Sunil Simon, editors, *FSTTCS 2020*, volume 182 of *LIPICs*, pages 56:1–56:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.FSTTCS.2020.56.
- 36 Géraud Sénizergues. $L(A)=L(B)$? decidability results from complete formal systems. *Theor. Comput. Sci.*, 251(1-2):1–166, 2001. doi:10.1016/S0304-3975(00)00285-1.
- 37 Richard Edwin Stearns and Harry B Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3):598–611, 1985.
- 38 Leslie G. Valiant. A note on the succinctness of descriptions of deterministic languages. *Inf. Control.*, 32(2):139–145, 1976. doi:10.1016/S0019-9958(76)90173-X.
- 39 Igor Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001. doi:10.1006/inco.2000.2894.

A Appendix

A.1 Resolvers with End-of-word Markers

As mentioned in the main part, GFG-PDA are by definition required to end their run with the last letter of the input word. Instead, one could also consider a model where they are allowed to take some trailing ε -transitions after the last input letter has been processed. As a resolver has access to the next input letter, which is undefined in this case, we need resolvers with end-of-word markers to signal the resolver that the last letter has been processed. In the following, we show that GFG-PDA with end-of-word resolvers are as expressive as standard GFG-PDA, albeit exponentially more succinct.

Fix some distinguished end-of-word-marker $\#$, which takes the role of the next input letter to be processed, if there is none after the last letter of the input word is processed. Let $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ be a PDA with $\# \notin \Sigma$. An EoW-resolver for \mathcal{P} is a function $r: \Delta^* \times (\Sigma \cup \{\#\}) \rightarrow \Delta$ such that for every $w \in L(\mathcal{P})$, there is an accepting run $c_0\tau_0 \cdots \tau_n c_n$ on w such that $\tau_{n'} = r(\tau_0 \cdots \tau_{n'-1}, w\#(|\ell(\tau_0 \cdots \tau_{n'-1})|))$ for all $0 \leq n' < n$. Note that the second argument given to the resolver is a letter of $w\#$, which is equal to $\#$ if the run prefix induced by $\tau_0 \cdots \tau_{n'-1}$ has already processed the full input w .

► **Lemma 19.** *GFG-PDA with EoW-resolvers are as expressive as GFG-PDA.*

Proof. A (standard) resolver can be turned into an EoW-resolver that ignores the EoW-marker. Hence, every GFG-PDA is a GFG-PDA with EoW-resolver recognizing the same language. So, it only remains to consider the other inclusion.

To this end, let $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ be a PDA with EoW-resolver. The language

$$C_{acc} = \{\gamma q \mid q \in F \text{ and } \gamma \in \perp\Gamma^*\} \subseteq \perp\Gamma^*Q$$

encoding final configurations of \mathcal{P} is regular. Hence, the language

$$C = \{\gamma q \in \perp\Gamma^*Q \mid \text{there is a run infix } (q, \gamma)\tau_0 \cdots \tau_{n-1}c_n \\ \text{with } \ell(\tau_0 \cdots \tau_{n-1}) = \varepsilon \text{ and } c_n \in C_{acc}\}$$

can be shown to be regular as well by applying saturation techniques [7]⁴ to the restriction of \mathcal{P} to ε -transitions. If \mathcal{P} reaches a configuration $c \in C$ after processing an input w , then $w \in L$, even if c 's state is not final.

Let $\mathcal{A} = (Q_{\mathcal{A}}, \Gamma_{\perp} \cup Q, q_I^{\mathcal{A}}, \delta_{\mathcal{A}}, F_{\mathcal{A}})$ be a DFA recognizing C . We extend the stack alphabet of \mathcal{P} to $\Gamma \times Q_{\mathcal{A}} \times (Q_{\mathcal{A}} \cup \{u\})$, where u is a fresh symbol. Then, we extend the transition relation such that it keeps track of the unique run of \mathcal{A} on the stack content: If \mathcal{P} reaches a stack content $\perp(X_1, q_1, q'_1)(X_2, q_2, q'_2) \cdots (X_s, q_s, q'_s)$, then we have

$$q_j = \delta_{\mathcal{A}}^*(q_I^{\mathcal{A}}, \perp X_1 \cdots X_j)$$

for every $1 \leq j \leq s$ as well as $q'_j = q'_{j-1}$ for every $2 \leq j \leq s$ and $q'_1 = u$. Here, $\delta_{\mathcal{A}}^*$ is the standard extension of $\delta_{\mathcal{A}}$ to words. The adapted PDA is still good-for-games, as no new nondeterminism has been introduced, and keeps track of the state of \mathcal{A} reached by processing the stack content as well as the shifted sequence of states of \mathcal{A} , which is useful when popping the top stack symbol: If the topmost stack symbol (X, q, q') is popped of the stack then q' is the state of \mathcal{A} reached when processing the remaining stack.

⁴ Also, see the survey by Carayol and Hague [8] for more details.

Now, we double the state space of \mathcal{P} , making one copy final, and adapt the transition relation again so that a final state is reached whenever \mathcal{P} would reach a configuration in C . Whether a configuration in C is reached can be determined from the current state of \mathcal{P} being simulated, as well as the top stack symbol containing information on the run of \mathcal{A} on the current stack content. The resulting PDA \mathcal{P}' recognizes $L(\mathcal{P})$ and has on every word $w \in L(\mathcal{P})$ an accepting run without trailing ε -transitions. Furthermore, an EoW-resolver for \mathcal{P} can be turned into a (standard) resolver for \mathcal{P}' , as the tracking of stack contents and the doubling of the state space does not introduce nondeterminism. ◀

As \mathcal{A} has at most exponential size, \mathcal{P}' is also exponential (both in the size of \mathcal{P}). This exponential blowup incurred by removing the end-of-word marker is in general unavoidable. In Theorem 10, we show that the language L_n of bit strings whose n^{th} bit from the end is a 1 requires exponentially-sized GFG-PDA. On the other hand, it is straightforward to devise a polynomially-sized GFG-PDA \mathcal{P}_{EoW} with EoW-marker recognizing L_n : the underlying PDA stores the input word on the stack, guesses nondeterministically that the word has ended, uses n (trailing) ε -transitions to pop of the last $n - 1$ letters stored on the stack, and then checks that the topmost stack symbol is a 1. With an EoW-resolver, the end of the input does not have to be guessed, but is marked by the EoW-marker. Hence, \mathcal{P}_{EoW} is good-for-games.

A.2 Pushdown Resolvers with Regular Stack Access

Recall that pushdown transducers implementing a resolver have access to the mode of the GFG-PDA whose nondeterminism it resolves. Here, we consider a more general model where the transducer can use information about the whole stack when determining the next transition. More precisely, we consider a regular abstraction of the possible stack contents by fixing a DFA running over the stack and allowing the transducer to base its decision on the state reached by the DFA as well.

Then, given a PDA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$, a pushdown resolver with regular stack access $\mathcal{T} = (\mathcal{D}, \mathcal{A}, \lambda)$ consists a DPDA \mathcal{P} with input alphabet Δ , a DFA \mathcal{A} over Γ_{\perp} with state set $Q^{\mathcal{A}}$, and an output function λ with output alphabet $Q \times Q^{\mathcal{A}} \times \Sigma \rightarrow \Delta$ such that the function $r_{\mathcal{T}}$ defined as follows, is a resolver for \mathcal{P} :

$$r_{\mathcal{T}}(\tau_0 \dots \tau_k, a) = \lambda(q_{\mathcal{T}})(q_{\mathcal{P}}, q_{\mathcal{A}}, a)$$

where

- $q_{\mathcal{T}}$ is the state of the last configuration of the longest run of \mathcal{D} on $\tau_0 \dots \tau_k$ (recall that while \mathcal{D} is deterministic, it may have several runs on an input which differ on trailing ε -transitions).
- Let c be the last configuration of the run of \mathcal{P} induced by $\tau_0 \dots \tau_k$. Then, $q_{\mathcal{P}}$ is the state of c and $q_{\mathcal{A}}$ is the state of \mathcal{A} reached when processing the stack content of c .

Every pushdown resolver with only access to the current mode is a special case of a pushdown resolver with regular stack access. On the other hand, having regular access to the stack is strictly stronger than having just access to the mode. However, by adapting the underlying GFG-PDA, one can show that the *languages* recognised by GFG-PDA with pushdown resolvers does not increase when allowing regular stack access.

► **Lemma 20.** *Every GFG-PDA with a pushdown resolver with regular stack access can be turned into an equivalent GFG-PDA with a pushdown resolver.*

53:20 A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct

Proof. Let $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, F)$ be a GFG-PDA and let $(\mathcal{D}, \mathcal{A}, \lambda)$ be a pushdown resolver with stack access for \mathcal{P} . We keep track of the state \mathcal{A} reaches on the current stack as in the proof of Lemma 19: If a stack content $\perp(X_1, q_1) \cdots (X_s, q_s)$ is reached, then q_j is the unique state of \mathcal{P} reached when processing $\perp X_1 \cdots X_j$. Now, it is straightforward to turn $(\mathcal{D}, \mathcal{A}, \lambda)$ into a pushdown resolver for \mathcal{P} that has only access to the top stack symbol. ◀

Perfect Forests in Graphs and Their Extensions

Gregory Gutin 

Royal Holloway, University of London, UK

Anders Yeo  

University of Southern Denmark, Odense, Denmark

University of Johannesburg, South Africa

Abstract

Let G be a graph on n vertices. For $i \in \{0, 1\}$ and a connected graph G , a spanning forest F of G is called an i -perfect forest if every tree in F is an induced subgraph of G and exactly i vertices of F have even degree (including zero). An i -perfect forest of G is proper if it has no vertices of degree zero. Scott (2001) showed that every connected graph with even number of vertices contains a (proper) 0-perfect forest. We prove that one can find a 0-perfect forest with minimum number of edges in polynomial time, but it is NP-hard to obtain a 0-perfect forest with maximum number of edges. We also prove that for a prescribed edge e of G , it is NP-hard to obtain a 0-perfect forest containing e , but we can find a 0-perfect forest not containing e in polynomial time. It is easy to see that every graph with odd number of vertices has a 1-perfect forest. It is not the case for proper 1-perfect forests. We give a characterization of when a connected graph has a proper 1-perfect forest.

2012 ACM Subject Classification Mathematics of computing \rightarrow Graph theory

Keywords and phrases graphs, odd degree subgraphs, perfect forests, polynomial algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.54

Funding *Gregory Gutin*: Research supported by the Leverhulme Trust under grant number RPG-2018-161.

Anders Yeo: Research supported by the Danish Research Foundation under grant number DFF 7014-00037B.

1 Introduction

In this paper all graphs are finite, undirected, have no parallel edges or loops. We use standard terminology and notation, see e.g. [5]. The number of vertices (edges, respectively) of a graph G is called its *order* (*size*, respectively). The degree of a vertex x in a graph G is denoted by $d_G(x)$. A vertex x of a graph G is a *cut-vertex* if $G - x$ has more connected components than G . A maximal connected subgraph of a graph G without a cut-vertex is called a *block*. Thus, every block of G is either a maximal 2-connected subgraph or a bridge (including its vertices) or an isolated vertex, implying that a block of odd order in a connected graph of order at least 3, must be a maximal 2-connected subgraph.

A spanning forest F of G is called a *semiperfect forest* if every tree of F is an induced subgraph of G . Let G be a graph and let $f: V(G) \rightarrow \{0, 1\}$ be a function such that $\sum_{v \in V(G)} f(v)$ is even (we will call such a function *even-sum*). A subgraph H in G where $d_H(x) \equiv f(x) \pmod{2}$ for all $x \in V(G)$, is called an *f -parity subgraph*. Note that the requirement that f is even-sum is necessary as otherwise an f -parity subgraph does not exist. An f -parity subgraph H of G is called an *f -parity perfect forest* if H is a semiperfect forest.

For $i \in \{0, 1\}$ and a graph G , an f -parity perfect forest is called an *i -perfect forest* if $f(x) = 1$ for all vertices of G for $i = 0$, and for all vertices of G apart from one for $i = 1$. An i -perfect forest of G is *proper* if it has no vertices of degree zero. Note that every 0-perfect forest (called a perfect forest in [3, 9] and a pseudo-matching in [18]) is proper. For examples of 0-perfect and 1-perfect forests, see Figures 1 and 2.



© Gregory Gutin and Anders Yeo;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

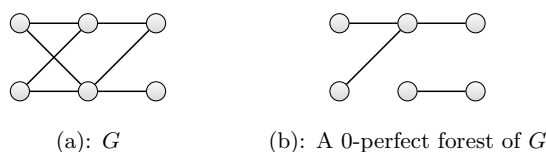
Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 54; pp. 54:1–54:13

Leibniz International Proceedings in Informatics

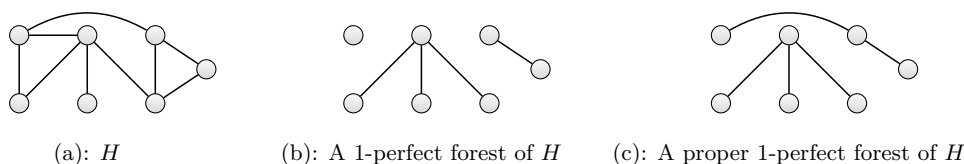


LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

54:2 Perfect Forests in Graphs and Their Extensions



■ **Figure 1** A graph G is shown in (a) and a 0-perfect forest of G is shown in (b) (as all degrees are odd and the trees are induced in G).



■ **Figure 2** The graph H is shown in (a), a (non-proper) 1-perfect forest of H is shown in (b), and a proper 1-perfect forest of H is shown in (c).

Clearly, every connected graph with a 0-perfect forest is of even order. Scott [17] proved that somewhat surprisingly the opposite implication is also true.

► **Theorem 1.** *Every connected graph of even order contains a 0-perfect forest.*

The proof of Theorem 1 in [17] is graph-theoretical and relatively long. A short proof using basic linear algebra is obtained in [9] and two short graph-theoretical proofs are given in [3]. All the proofs of Theorem 1 are constructive and yield polynomial algorithms for finding 0-perfect forests. Intuitively, it is clear that a 0-perfect forest can provide a useful structure in a graph and, in particular, this notion was used by Sharan and Wigderson [18] to prove that the perfect matching problem for bipartite cubic graphs belongs to the complexity class \mathcal{NC} . Semiperfect forests were used in the proofs of three theorems in [7]. Gutin and Yeo [11] studied extensions of a 0-perfect forest to directed graphs.

Since a 0-perfect forest is a generalization of a matching, it is natural to study the following two problems for a connected graph G of even order n :

- (1) Find a 0-perfect forest of G of minimum size. (Clearly, the minimum size is $n/2$ if and only if G has a perfect matching.)
- (2) Find a 0-perfect forest of G of maximum size. (This is of interest in matching-like edge-decompositions of G .)

The following theorem solves the first problem.

► **Theorem 2.** *In polynomial time, we can find a 0-perfect forest of minimum size.*

Theorem 2 follows immediately from the next theorem by letting $f(x) = 1$ for all $x \in V(G)$. Theorem 3 shows usefulness of extending Problem 1 to f -parity perfect forests. Theorem 3 is proved in Section 2.

► **Theorem 3.** *Let G be a connected graph and let $f: V(G) \rightarrow \{0, 1\}$ be an even-sum function. We can in polynomial time find an f -parity perfect forest H in G , such that $d_H(x) \equiv f(x) \pmod{2}$ for all $x \in V(G)$ and $|E(H)|$ is minimized.*

As the following theorem shows, the second problem cannot be solved in polynomial time unless $P=NP$.

► **Theorem 4.** *It is NP-hard to find a 0-perfect forest of maximum size.*

Let $n = |V(G)|$. Theorem 4 follows from the next result proved in Section 3. Theorem 5 is optimal in the following sense. The problem of finding a 0-perfect forest of size at least $n - 1$ is polynomial-time solvable because G has a 0-perfect forest of size at least $n - 1$ if and only if G is a tree in which every vertex is of odd degree.

► **Theorem 5.** *It is NP-hard to decide whether a connected graph contains a 0-perfect forest with at least $n - 2$ edges.*

It is easy to show that Theorem 5 holds if we replace $n - 2$ by $n - k$ for any integer $k \geq 2$. Indeed, add two new vertices x and y to a graph G as well as two edges xy and yu , where u is any vertex in G . The resulting graph is denoted by G' . Observe that there is a 0-perfect forest of size $|V(G)| - k$ in G if and only if there is a 0-perfect forest of size $|V(G')| - (k + 1)$ in G' .

Since the problem of finding a 0-perfect forest of maximum size is NP-hard, it is natural to study its parameterized complexity using appropriate parameterizations e.g. the parameterization below the tight upper bound $n - 1$ and the parameterization above the tight upper bound $n/2$. In other words, we can ask whether there is a 0-perfect forest of size at least $n - k$ ($n/2 + k$, respectively), where k is the parameter. (Above-tight-lower-bound and below-tight-upper-bound parameterizations were studied for many graph-theoretical and constraint satisfaction problems, see e.g. [1, 4, 10, 13, 14].) Theorem 5 shows that the parameterization $n - k$ is **para-NP-complete** (for an introduction to **para-NP-completeness**, see e.g. [6]). We do not know the answer to the following question. Is the parameterization $n/2 + k$ fixed-parameter tractable?¹

Here is another pair of natural problems on 0-perfect forests. They both are clearly polynomial-time solvable when restricted to perfect matchings. For a graph G of even order and an edge e in G ,

- (1') find a 0-perfect forest containing e ;
- (2') find a 0-perfect forest not containing e .

For Problem 1', we prove the following result in Section 4.

► **Theorem 6.** *The following problem is NP-hard. Given a connected graph G and an edge $e \in E(G)$, decide whether G has a 0-perfect forest containing e .*

For Problem 2', we have the next result, which follows immediately from Theorem 8, by letting $f(x) = 1$ for all x in G . Theorem 8 again demonstrates usefulness of f -parity perfect forests. It is proved in Section 5.

► **Theorem 7.** *Given a graph G and an edge $e \in E(G)$ we can in polynomial time decide whether G has a 0-perfect forest not containing e .*

► **Theorem 8.** *The following problem is polynomial time solvable. Given a graph G , an edge $e \in E(G)$ and an even-sum function $f: V(G) \rightarrow \{0, 1\}$, decide whether G has an f -parity perfect forest not containing e .*

¹ While working on the final version of this paper, we obtained a proof that the parameterized problem is W[1]-hard. We will include the proof in a journal version of the paper.

Since an odd order connected graph cannot have a 0-perfect forest, it is natural to ask whether every connected graph of odd order has a 1-perfect forest (recall that a 1-perfect forest has only one vertex of even degree). The answer is positive and the proof is trivial. In fact, it is not hard to show the following strengthening of this observation, which will be useful in the proof of Theorem 10.

► **Proposition 9.** *Let x be an arbitrary vertex of a connected graph G of odd order. Then G has a 1-perfect forest F such that $d_F(x)$ is even.*

Proof. Create a new graph H by adding a new vertex y to G and adding the edge xy . By Theorem 1, H has a 0-perfect forest, F_H . Deleting the vertex y from F_H , results in the desired 1-perfect forest of G where x is the only vertex of even degree. ◀

Note that not every connected graph of odd order has a proper 1-perfect forest. For example, no complete graph of odd order has such a forest. Thus, a more interesting question with a potentially more useful answer is when a connected graph of odd order has a proper 1-perfect forest? This question is answered in the following characterization proved in Section 6.

► **Theorem 10.** *Let \mathcal{B} be the set of all connected graphs where every block is a complete graph of odd order. If G is a connected graph of odd order $n \geq 3$ then G contains a proper 1-perfect forest if and only if $G \notin \mathcal{B}$.*

Using this theorem and a linear-time algorithm for computing biconnected components in a graph [12], in polynomial time we can decide whether a connected graph G of odd order contains a proper 1-perfect forest. If $G \notin \mathcal{B}$, the proof by induction of Theorem 2 yields a polynomial-time recursive algorithm to construct a proper 1-perfect forest.

Our proof of Theorem 10 is graph-theoretical and so are the proofs of Theorem 1 in [17] and [3]. Recall that Gutin [9] gave a linear-algebraic proof of Theorem 1. It would be interesting to see whether Theorem 10 can be proved using a linear-algebraic approach, too.

2 Proof of Theorem 3

► **Lemma 11.** *Let G be a connected graph and let $f: V(G) \rightarrow \{0, 1\}$ be an even-sum function. If H is an f -parity subgraph of G of minimum size, then H is an f -parity perfect forest.*

Proof. Assume that H is an f -parity subgraph with minimum possible $|E(H)|$. Clearly H contains no cycles, as removing the edges of a cycle would contradict the minimality of $|E(H)|$. Assume that some tree T of H is not an induced tree in G . Let xy be an edge of G , not belonging to T but with $\{x, y\} \subseteq V(T)$. Remove the unique (x, y) -path in T from H and add the edge xy to H . This decreases the number of edges in H without changing the parity of the degree of any vertex, contradicting the minimality of $|E(H)|$. Therefore H is indeed an f -parity perfect forest. ◀

Lemma 11 implies the following:

► **Theorem 12.** *Let G be a connected graph and let $f: V(G) \rightarrow \{0, 1\}$ be an even-sum function. Then there exists an f -parity perfect forest F in G .*

Proof. Let $x_1, x_2, \dots, x_k, y_1, y_2, \dots, y_k$ be the vertices in G with f -value equal to one. Let P_i be any (x_i, y_i) -path in G for all $i = 1, 2, \dots, k$, which exists as G is connected. Let H be the spanning subgraph of G such that an edge $e \in E(G)$ belongs to H if and only if e

belongs to an odd number of paths in P_1, P_2, \dots, P_k . Let $x \in V(G)$. Observe that $d_H(x)$ is odd if and only if x is incident with an odd number of edges in $\cup_{i=1}^k E(P_i)$, which is if and only if x is the endpoint of one of the paths i.e. $f(x) = 1$. Thus, H is an f -parity subgraph of G . Lemma 11 now implies that if H is the f -parity subgraph of G of minimum size, then H is an f -parity perfect forest. \blacktriangleleft

Note that Theorem 12 generalizes Theorem 1: set $f(x) = 1$ for all $x \in V(G)$. Thus, Theorem 12 provides an alternative proof of Theorem 1.

► Theorem 3. *Let G be a connected graph and let $f: V(G) \rightarrow \{0, 1\}$ be an even-sum function. We can in polynomial time find an f -parity perfect forest H in G , such that $d_H(x) \equiv f(x) \pmod{2}$ for all $x \in V(G)$ and $|E(H)|$ is minimized.*

Proof. Let G be a connected graph and let $f: V(G) \rightarrow \{0, 1\}$ be an even-sum function. Let $V(G) = \{v_1, v_2, \dots, v_n\}$. We will construct a weighed auxillary graph H as follows. Let $V(H) = \cup_{i=1}^n X_i$, where for every $i \in [n]$, $|X_i| \in \{n-1, n\}$ and $|X_i| \equiv f(v_i) \pmod{2}$. For all $1 \leq i < j \leq n$ and all $u \in X_i$ and $v \in X_j$, we let $uv \in E(H)$ if and only if $v_i v_j \in E(G)$. Finally add a matching $M_i = \{e_1^i, e_2^i, \dots, e_{\lfloor |X_i|/2 \rfloor}^i\}$ to X_i for all $i \in [n]$. Let the weight of all the edges within each X_i (i.e. the edges in M_i) be zero and let all edges between different X_i 's have weight one.

We first show that H contains a perfect matching. As $\sum_{v \in V(G)} f(v)$ is even we may assume that $\{v_1, v_2, \dots, v_{2k}\}$ are the vertices of G with an f -value of one for some integer k with $0 \leq k \leq n/2$. Assume that $y_i \in X_i$ is the unique vertex in X_i that is not saturated by M_i for all $i \in [2k]$ and start of by letting M be the matching containing all M_i 's.

Let $P_i = v_i v_{p_1^i} v_{p_2^i} \dots v_{p_{i-1}^i} v_{i+k}$ be any path in G from v_i to v_{i+k} where $i \in [k]$. It is not difficult to see that there exists an M -augmenting path, Q_i , in H starting in y_i and ending in y_{i+k} and containing exactly the edges $e_{i-1}^{p_1^i}, e_{i-2}^{p_2^i}, \dots, e_i^{p_{i-1}^i}$ from M . Also observe that Q_1, Q_2, \dots, Q_k are vertex disjoint, which implies that we can use all Q_i to increase the matching M thereby obtaining a perfect matching in H .

We will now show the following claim. The *size* of a multiset S is the total number of elements in S , where if an element $e \in S$ is of multiplicity r , then e is counted r times.

▷ **Claim A.**

- (a) If there exists a perfect matching in H with weight w^* then there exists a multiset of edges E^* in G of size w^* , such that $d_{E^*}(x) \equiv f(x) \pmod{2}$ for all $x \in V(G)$.
- (b) Conversely if E^* is a multiset of edges in G of size w^* , such that $d_{E^*}(x) \equiv f(x) \pmod{2}$ for all $x \in V(G)$, then there exists a perfect matching in H with weight at most w^* .

Proof of Claim A. First assume that we have a multiset of edges E^* in G of size $w^* \leq W_{\max}$, such that $d_{E^*}(x) \equiv f(x) \pmod{2}$ for all $x \in V(G)$. Let $M^* = \emptyset$. For every $v_i v_j \in E^*$ we will add edges between X_i and X_j to M^* as follows: if $v_i v_j$ is of multiplicity r in E^* , then we add an edge between X_i and X_j to M^* if and only if r is odd. Since we will add $2k_i + f(v_i)$ edges that are incident to X_i for each $i \in [n]$ (where k_i is some integer), we can add these edges such that their endvertices are $V(e_1^i) \cup V(e_2^i) \cup \dots \cup V(e_{k_i}^i)$ if $f(v_i) = 0$ and $\{y_i\} \cup V(e_1^i) \cup V(e_2^i) \cup \dots \cup V(e_{k_i}^i)$ if $f(v_i) = 1$ for each $i \in [n]$, where $V(e_j^i)$ denotes the pair of endvertices of e_j^i . We can now extend M^* to a perfect matching by adding $M_i \setminus \{e_1^i, e_2^i, \dots, e_{k_i}^i\}$ for each $i \in [n]$. This gives us a perfect matching in H with weight at most $|E^*|$ as desired.

Conversely assume that there exists a perfect matching M^* in H with weight w^* . Initially let $E^* = \emptyset$. For every $xy \in M^*$ with weight one (i.e. $x \in X_i$ and $y \in X_j$ for some $i \neq j$), add $v_i v_j$ to E^* . This gives us the desired multiset E^* , thereby completing the proof of Claim A. \blacktriangleleft

We have proved that H has a perfect matching. Let M_{\min} be a minimum weight perfect matching in H which can be determined in polynomial time using Edmonds' blossom algorithm as a subroutine, see e.g. [15]. Let W_{\min} be the weight of M_{\min} . By Claim A(a), using M_{\min} , in polynomial time we can find a multiset of edges E^* in G of size W_{\min} , such that $d_{E^*}(x) \equiv f(x) \pmod{2}$ for all $x \in V(G)$. By Claim A(b), since W_{\min} is the minimum weight of a perfect matching in H , W_{\min} is minimum size of a multiset of edges E^{**} , such that $d_{E^{**}}(x) \equiv f(x) \pmod{2}$ for all $x \in V(G)$.

Note that no edge is in E^* more than once, since if some edge, e , appears twice, then we can delete two copies of e from E^* , thereby contradicting the minimality of $|E^*|$. Let F be the spanning subgraph of G with edge set E^* . By Lemma 11 we note that F is an f -parity perfect forest, which completes the proof of the theorem. ◀

3 Proof of Theorem 5

We will reduce from the *not-all-equal 3-SAT* problem, abbreviated to NAE-3-SAT, which is the problem of determining whether an instance of 3-SAT has a truth assignment to its variables such that every clause contains both a true and a false literal. If this is the case we say that the instance is *NAE-satisfied*. NAE-3-SAT is known to be NP-hard to solve [16]. Let I be an instance of NAE-3-SAT with clauses C_1, C_2, \dots, C_m and variables v_1, v_2, \dots, v_n . We will construct a graph G such that G contains a 0-perfect forest with at least $n - 2$ edges if and only if I is NAE-satisfied.

We first create a gadget H_i for each $i = 1, 2, \dots, n$ as follows. Let

$$V(H_i) = \{x_1^i, z_1^i, y_1^i, x_2^i, z_2^i, y_2^i\}$$

and add all possible edges to H_i , except $x_1^i y_1^i$ and $x_2^i y_2^i$. For all $i = 1, 2, \dots, n - 1$ we then add all edges between $\{y_1^i, y_2^i\}$ and $\{x_1^{i+1}, x_2^{i+1}\}$. Now add a pendent edge to each vertex in $V(H_i) \setminus \{x_1^i, x_2^i, y_1^i, y_2^i\}$ for all $i = 1, 2, \dots, n$. See Figure 3 for an illustration of this part of G , which is denoted by Q . We will now complete our construction of G .

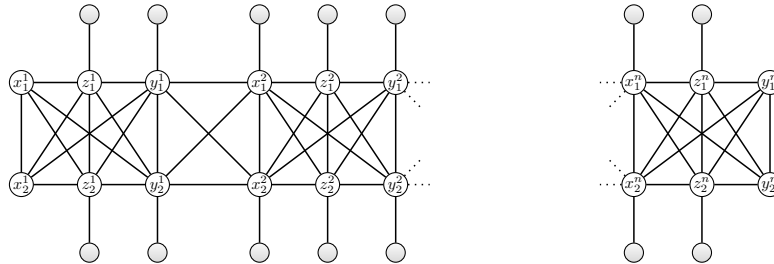
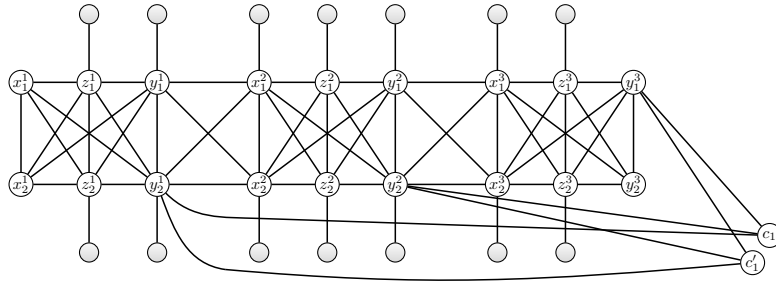


Figure 3 The gadgets H_1, H_2, \dots, H_n and the edges connecting these. The resulting graph is denoted by Q .

Let $V(G) = V(Q) \cup \{c_1, c_2, \dots, c_m\} \cup \{c'_1, c'_2, \dots, c'_m\}$. For each $j = 1, 2, \dots, m$ we will add an edge from both c_j and c'_j to y_2^i if and only if v_i is a literal in the clause C_j . We will furthermore add an edge from both c_j and c'_j to y_1^i if and only if \bar{v}_i is a literal in the clause C_j . This completes the construction of G . See Figure 4 depicting G for $I = (v_1, v_2, \bar{v}_3)$.

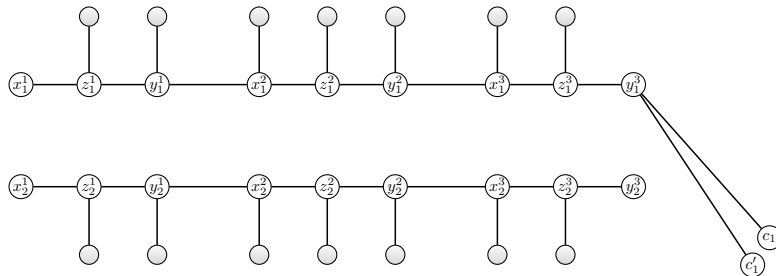
We will now show that G contains a 0-perfect forest of size at least $n - 2$ if and only if I is NAE-satisfied. First assume that I is NAE-satisfied and consider a truth assignment τ NAE-satisfying I . We will construct two vertex-disjoint induced trees, T_1 and T_2 , in G , such that all degrees in the trees T_i are odd for $i \in [2]$. If v_i is true in τ then add the vertices in



■ **Figure 4** The graph G if $I = (v_1, v_2, \bar{v}_3)$.

$\{x_1^i, z_1^i, y_1^i\}$ to T_1 and the vertices in $\{x_2^i, z_2^i, y_2^i\}$ to T_2 . Conversely, if v_i is false in τ then add the vertices in $\{x_1^i, z_1^i, y_1^i\}$ to T_2 and the vertices in $\{x_2^i, z_2^i, y_2^i\}$ to T_1 . We furthermore add all vertices of degree one to the same tree as their neighbour. Note that the vertices we have added so far to T_i (for $i \in [2]$) induce a tree in G , where every vertex has odd degree in T_i .

Finally as I is NAE-satisfied we note for $j \in [m]$, each of c_j and c'_j has one edge into one of the T_i 's and two edges into the other T_i . Add each of c_j and c'_j to the T_i with which it is only connected by one edge. We note that after this operation the vertices we have added so far to T_i (for $i \in [2]$) still induces a tree in G where every vertex has odd degree in T_i . After doing the above operation for all $j \in [m]$ we have obtained the desired trees T_1 and T_2 whose union form a 0-perfect forest in G with $|V(G)| - 2$ edges. See Figure 5 for the found T_1 and T_2 if the instance of NAE-3-SAT is $I = (v_1, v_2, \bar{v}_3)$ and the truth assignment is to set all variables equal to true.



■ **Figure 5** The trees T_1 and T_2 if $I = (v_1, v_2, \bar{v}_3)$ and $v_1 = v_2 = v_3 = \text{true}$.

Conversely, assume that G contains a 0-perfect forest with at least $|V(G)| - 2$ edges. As G is not a tree this implies that G contain two vertex-disjoint trees T_1 and T_2 such that each T_i is an induced tree in G of order at least 2, all degrees in each T_i are odd, and $V(T_1)$ and $V(T_2)$ partition $V(G)$. We will now prove the following claims where Claim C completes the proof of the theorem.

▷ **Claim A.** For each $i \in [n]$ one of the following cases hold.

- A.1:** $\{x_1^i, z_1^i, y_1^i\} \in V(T_1)$ and $\{x_2^i, z_2^i, y_2^i\} \in V(T_2)$.
- A.2:** $\{x_1^i, z_2^i, y_1^i\} \in V(T_1)$ and $\{x_2^i, z_1^i, y_2^i\} \in V(T_2)$.
- A.3:** $\{x_1^i, z_1^i, y_1^i\} \in V(T_2)$ and $\{x_2^i, z_2^i, y_2^i\} \in V(T_1)$.
- A.4:** $\{x_1^i, z_2^i, y_1^i\} \in V(T_2)$ and $\{x_2^i, z_1^i, y_2^i\} \in V(T_1)$.

Proof of Claim A. As the only two non-edges in H_i are $x_1^i y_1^i$ and $x_2^i y_2^i$ we note that there exist a 4-cycle on every set of 4 vertices in H_i . Therefore $|V(T_j) \cap V(H_i)| \geq 4$ is not possible for any $j \in [2]$ and $i \in [n]$. So $|V(T_j) \cap V(H_i)| = 3$ for $j \in [2]$ and $i \in [n]$.

As there is no 3-cycle in $G[V(T_j)]$ for $j \in [2]$ we note that x_1^i and y_1^i must belong to one of the trees, say T_j , and x_2^i and y_2^i must belong to the other tree, T_{3-j} . So if $x_1^i \in V(T_1)$ then $y_1^i \in V(T_1)$ and $\{x_2^i, y_2^i\} \subseteq V(T_2)$ and we are in case A.1 or A.2. On the other hand if $x_1^i \in V(T_2)$ then $y_1^i \in V(T_2)$ and $\{x_2^i, y_2^i\} \subseteq V(T_1)$ and we are in case A.3 or A.4. This completes the proof of Claim A. \triangleleft

\triangleright **Claim B.** For $i = 1, 2$, $G[V(Q) \cap V(T_i)]$ is a tree where all vertices have odd degree.

Proof of Claim B. Any vertex in G with degree one must belong to the same tree, T_j , as its neighbour, as both T_1 and T_2 have order at least two. By Claim A, we therefore note that $G[V(Q) \cap V(T_i)]$ is a path of length $3n$ with a pendent edge attached to each non-endpoint of the path. This implies that $G[V(Q) \cap V(T_i)]$ is a tree where all vertices have odd degree (as all degrees are either 1 or 3). This completes the proof of Claim B. \triangleleft

\triangleright **Claim C.** The instance I is NAE-satisfiable.

Proof of Claim C. Assume that the vertex c_j belongs to T_1 . First suppose that $|N_G(c_j) \cap V(T_1)| = 0$. In this case c_j has no neighbours in T_1 , a contradiction, as T_1 is a tree with order at least two. So $|N_G(c_j) \cap V(T_1)| \geq 1$. Assume that $|N_G(c_j) \cap V(T_1)| \geq 2$. As T_1 is an induced tree in G , c_j must have at least two neighbours, say x and y , in T_1 . However, by Claim B, there exists a (x, y) -path in T_1 using only vertices from $V(Q)$, which implies that there is a cycle in T_1 , a contradiction. Therefore $|N_G(c_j) \cap V(T_1)| = 1$.

Analogously, we can show that $|N_G(c_j) \cap V(T_2)| = 1$, whenever $c_j \in V(T_2)$. So each clause C_j ($j \in [m]$) has either exactly one literal that is false (if $c_j \in V(T_1)$) or exactly one literal that is true (if $c_j \in V(T_2)$). This implies that I is NAE-satisfiable, which completes the proof of Claim C and the theorem. \triangleleft

4 Proof of Theorem 6

To prove Theorem 6, we will use the following result. The proof of Theorem 4 follows the same approach as the proof that it is NP-hard to determine whether there is an induced cycle of odd length through a prescribed vertex, given in [2] by Bienstock. The proof is not given here but can be found in the appendix of [8].

\blacktriangleright **Theorem 4.** *It is NP-hard to determine whether a graph contains an induced cycle through two given edges.*

\blacktriangleright **Theorem 6.** *The following problem is NP-hard. Given a connected graph G and an edge $e \in E(G)$, decide whether G has a 0-perfect forest containing e .*

Proof. Let G be a graph and let $e_1 = u_1v_1$ and $e_2 = u_2v_2$ be distinct edges of G . We will construct an auxiliary graph H with an edge $e'_2 \in E(H)$, such that H contains a 0-perfect forest containing e'_2 if and only if G contains an induced cycle, C , such that $e_1, e_2 \in E(C)$. This will complete the proof by Theorem 4.

Let H be obtained from G by adding a pendent edge to each vertex in $V(G) \setminus \{u_1, v_1\}$ and deleting the edge e_1 . Let E_P denote the set of all the pendent edges we just added to G . Let $e'_2 = u_2v_2$ and note that $e'_2 \in E(H)$. This completes the construction of H .

Assume that there exists an induced cycle, C , in G such that $e_1, e_2 \in E(C)$. Let $E' = E_P \cup E(C) \setminus e_1$. Note that the edges in E' induce a 0-perfect forest in H containing the edge e'_2 .

Conversely assume that there is a 0-perfect forest, F , in H containing e'_2 . Clearly F contains all edges in E_P as each pendent edge is incident with a vertex of degree one. Let Q be the subgraph of H induced by the edges in $E(F) \setminus E_P$. Note that Q is a perfect forest

where u_1 and v_1 have odd degree and all other vertices have even degree. As Q is a perfect forest all components are induced trees, and as u_1 and v_1 are the only vertices of odd degree, this implies that Q is an induced path between u_1 and v_1 . Adding the edge e_1 to Q gives us an induced cycle in G containing both e_1 and e_2 (as $e'_2 \in E(F)$).

Therefore we have proven that H contains a 0-perfect forest containing e'_2 if and only if G contains an induced cycle, C , such that $e_1, e_2 \in E(C)$, as desired. ◀

5 Proof of Theorem 8

Let G be a graph and $e = uv$ an edge of G . Let $f: V(G) \rightarrow \{0, 1\}$ be an even-sum function. Our polynomial-time algorithm will follow from the four claims proved below. At the end of the proof, we briefly discuss how the claims are used in the algorithm.

▷ **Claim A.** Suppose that G contains a cut-vertex x , which may or may not belong to $\{u, v\}$. Let C be the component in $G - x$ intersecting $\{u, v\}$ (there is exactly one such component as $uv \in E(G)$) and let $G' = G[V(C) \cup \{x\}]$. Let $f'(w) = f(w)$ for all $w \in V(C)$ and define $f'(x) \in \{0, 1\}$ such that $\sum_{z \in V(G_i)} f'(z)$ is even. Then G has an f -parity perfect forest not containing e if and only if G' has an f' -parity perfect forest not containing e .

Proof of Claim A. Let G contain a cut-vertex x and let C_1, C_2, \dots, C_k be the components in $G - x$. Without loss of generality, assume that C_1 is the component intersecting $\{u, v\}$. Let $G_i = G[V(C_i) \cup \{x\}]$ for all $i \in [k]$.

For each $i \in [k]$ we will let $f_i: V(G_i) \rightarrow \{0, 1\}$ be defined such that $f_i(w) = f(w)$ for all $w \in V(C_i)$ and $\sum_{z \in V(G_i)} f_i(z)$ is even (this defines the value of $f_i(x)$). We will show that G has an f -parity perfect forest not containing e if and only if G_1 has an f_1 -parity perfect forest not containing e , which will complete the proof of Claim A.

First assume that G_1 has an f_1 -parity perfect forest F_1 not containing e . By Theorem 12 there exists an f_i -parity perfect forest, F_i , in G_i for all $i = 2, 3, \dots, k$. Now $F_1 \cup F_2 \cup \dots \cup F_k$ is an f -parity perfect forest of G not containing e , as desired.

Conversely assume that G has an f -parity perfect forest F not containing e . If we restrict F to $V(G_1)$, then we obtain an f_1 -parity perfect forest of G_1 not containing e . ◀

▷ **Claim B.** If G is 2-connected and $f(u) = 0$ or $f(v) = 0$ then G has an f -parity perfect forest not containing e .

Proof of Claim B. Assume without loss of generality that $f(u) = 0$. As G is 2-connected $G - u$ is connected and $\sum_{z \in V(G-u)} f(z)$ is even. Therefore, by Theorem 12, there exists an f -parity perfect forest in $G - u$, which is also an f -parity perfect forest in G not containing the edge e . ◀

▷ **Claim C.** If G is 2-connected and $f(u) = f(v) = 1$ then G has a f -parity perfect forest if and only if $\sum_{z \in V(G)} f(z) \geq 4$.

Proof of Claim C. Let $S = \sum_{z \in V(G)} f(z)$. As f is even-sum, S is even. Since $f(u) = f(v) = 1$, we have $S \geq 2$. If $S = 2$ and F is an f -parity perfect forest in G , then u and v must be leaves of the same tree in F (as they are the only vertices with an f -value of one). Therefore $e \in E(F)$, as otherwise the tree containing u and v is not induced in G . So, if $S = 2$ then G has no f -parity perfect forest F in G with $e \notin E(F)$.

We may therefore assume that $S \geq 4$ and let $w \in V(G) \setminus \{u, v\}$ have $f(w) = 1$. As G is 2-connected there exists a (u, v) -path, P , in G with $w \in V(P)$. (To see it, consider two internally disjoint paths from w to w' where w' is a new vertex added to G such that

54:10 Perfect Forests in Graphs and Their Extensions

$N(w') = \{u, v\}$.) We now create a spanning tree T in G , such that $E(P) \subseteq E(T)$ and $d_T(w) = 2$, as follows. Initially let $T = P$. While $V(T) \neq V(G)$ let $q \in V(G) \setminus V(T)$ be arbitrary such that q has an edge into $V(T) \setminus \{w\}$ (which exists as G is 2-connected). Add q and an edge from q into $V(T) \setminus \{w\}$ to T . When $V(T)$ becomes equal to $V(G)$ we have our desired tree T .

Let T_1 and T_2 be the two trees in $T - w$ (there are exactly two trees in $T - w$ as $d_T(w) = 2$). Let $S_1 = \sum_{z \in V(T_1)} f(z)$ and let $S_2 = \sum_{z \in V(T_2)} f(z)$. As $f(w) = 1$ and $V(T_1) \cup V(T_2) = V(G) \setminus \{w\}$, we note that $S_1 + S_2$ is odd. If S_i is odd then add w to T_i ($i \in [2]$), using the edge from w to $V(T_i)$ in T . This results in two trees, say T'_1 and T'_2 , where $\sum_{z \in V(T'_i)} f(z)$ is even for $i \in [2]$. Furthermore, as $w \in V(P)$ and $E(P) \subseteq E(T)$, we note that u and v do not belong to the same tree T'_i . By Theorem 12 there exists an f -parity perfect forest, F'_i , of $G[V(T'_i)]$ for $i \in [2]$ (as T'_i is a spanning tree in $G[V(T'_i)]$, $G[V(T'_i)]$ is connected). Now $F'_1 \cup F'_2$ is an f -parity perfect forest of G not containing e . This completes the proof of Claim C. \triangleleft

It is easy to see that the following algorithm is of polynomial time. Keep reducing the graph (see Claim A) as long as there exists a cut-vertex and when there are no more cut-vertices then the answer is “no” if the endpoints of e have an f -value of one and all other vertices have an f -value of zero and “yes”, otherwise (see Claims B and C). See Figure 6 for an illustration of the algorithm.

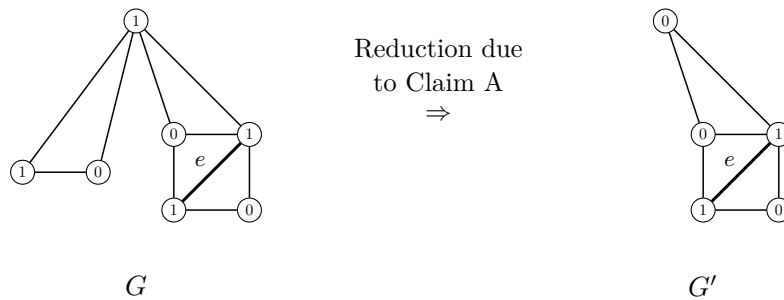


Figure 6 An illustration of the algorithm given in Theorem 8, where the values on the nodes indicate the f -values. As in the final graph the endpoints of e have an f -value of one and all other vertices have an f -value of zero there is no f -parity perfect forest in G' avoiding the edge e and therefore not in G either.

6 Proof of Theorem 10

Theorem 10 follows from Theorem 2 and Lemma 3 proved in this section. To prove Theorem 2, we will use the following:

► **Lemma 4.** *Let G be a connected graph of even order and let $xy \in E(G)$ such that $G - \{x, y\}$ is connected. If $G - x \in \mathcal{B}$ and $G - y \in \mathcal{B}$ then $N[x] = N[y]$.*

Proof. Let G be a connected graph of even order and let $xy \in E(G)$ be chosen such that $G - \{x, y\}$ is connected. Let $G_y = G - x$ and let $G_x = G - y$ and assume that $G_y \in \mathcal{B}$ and $G_x \in \mathcal{B}$. Let $C_1^x, C_2^x, \dots, C_{l_x}^x$ be the blocks of G_x and without loss of generality assume that $x \in V(C_1^x)$. Analogously, let $C_1^y, C_2^y, \dots, C_{l_y}^y$ be the blocks of G_y and without loss of generality assume that $y \in V(C_1^y)$.

▷ **Claim A.** $N_{G_x}[x] = V(C_1^x)$ and C_1^x is a complete graph of odd order and $C_1^x - x$ is a block in $G - \{x, y\}$. Analogously, $N_{G_x}[y] = V(C_1^y)$ and C_1^y is a complete graph of odd order and $C_1^y - y$ is a block in $G - \{x, y\}$.

Proof of Claim A. For the sake of contradiction assume that $u_1, u_2 \in N_{G_x}(x)$ but u_1 and u_2 belong to different blocks of G_x . In this case there is a cut-vertex in G_x separating u_1 and u_2 , which must be x (as u_1xu_2 is a path in G_x). However x does not separate u_1 and u_2 as $G - \{x, y\}$ is connected. This contradiction implies that all vertices in $N_{G_x}(x)$ belong to the same block of G_x .

Therefore, $N_{G_x}[x] \subseteq V(C_1^x)$ as x is not a cut-vertex in G_x (as $G - \{x, y\}$ is connected) and hence x only belongs to one block of G_x . As $G_x \in \mathcal{B}$ we note that C_1^x is a complete graph of odd order. As $|V(C_1^x)| \geq 3$ (as all blocks contain at least two vertices, and $|V(C_1^x)|$ is odd) and x is not a cut-vertex in G_x we note that $C_1^x - x$ is a block in $G - \{x, y\}$. This completes the proof of Claim A. ◁

We now return to the proof of the lemma. By Claim A we note that $C_1^y - y$ is a block in $G - \{x, y\}$ which furthermore is a complete graph of even order. If $C_1^x - x$ and $C_1^y - y$ are different blocks in $G - \{x, y\}$, then $C_1^y - y$ is a block of even order in G_x , a contradiction to $G_x \in \mathcal{B}$. So, $C_1^x - x$ and $C_1^y - y$ are the same block in $G - \{x, y\}$. By Claim A, we have the following chain of equalities, which completes the proof of the lemma.

$$N_G[x] = V(C_1^x - x) \cup \{x, y\} = V(C_1^y - y) \cup \{x, y\} = N_G[y] \quad \blacktriangleleft$$

► **Theorem 2.** *Every connected graph, $G \notin \mathcal{B}$, of odd order $n \geq 3$ contains a proper 1-perfect forest.*

Proof. The proof is by induction over odd integers $n \geq 3$. For $n = 3$, we have $G \cong P_3$, the path of order 3, which is a proper 1-perfect forest. Now we assume that G is a connected graph of odd order $n \geq 5$ such that $G \notin \mathcal{B}$. Let us consider two cases.

Case 1: G is not 2-connected. Assume that G has a cut-vertex x such that $G - x$ has a component C_1 of even order. Let $G_1 = G[V(C_1) \cup \{x\}]$ and let $G_2 = G - V(C_1)$. Note that both G_1 and G_2 are connected graphs of odd order. Furthermore the set of blocks of G is exactly the union of the blocks in G_1 and G_2 . As $G \notin \mathcal{B}$ (and therefore some block in G is not a complete graph of odd order) we note that either $G_1 \notin \mathcal{B}$ or $G_2 \notin \mathcal{B}$ (or both).

Let $i \in \{1, 2\}$ be defined such that $G_i \notin \mathcal{B}$ and let $j = 3 - i$. By induction hypothesis, there exists a proper 1-perfect forest F_i in G_i . By Theorem 9 there also exists a (not necessarily proper) 1-perfect forest, F_j , in G_j , where x is the vertex of even degree in F_j . We now note that $F_i \cup F_j$ is a proper 1-perfect forest of G , where the only vertex of even degree is the vertex of even degree in F_i . Thus, we may assume that G has no cut-vertex x such that some component in $G - x$ is of even order.

Now assume that G contains a cut-vertex x . By the previous assumption, all components in $G - x$ are of odd order, and let C_1 be a component of $G - x$. Let $G_1 = G[V(C_1) \cup \{x\}]$ and let $G_2 = G - V(C_1)$. Note that both G_1 and G_2 are connected graphs of even order. By Theorem 1 there exists a 0-perfect forest F_1 in G_1 and a 0-perfect forest F_2 in G_2 . Note that $F_1 \cup F_2$ is now a proper 1-perfect forest of G , where the only vertex of even degree is x .

Case 2: G is 2-connected.

► **Definition A.** As $G \notin \mathcal{B}$ and G has odd order, we note that G is not a complete graph. Therefore there exists an induced path $p_1p_2p_3$ in G (that is, $p_1p_2, p_2p_3 \in E(G)$ and $p_1p_3 \notin E(G)$). Let C_1, C_2, \dots, C_l be the components in $G - \{p_2, p_3\}$, such that $p_1 \in C_1$.

Assume first that $|V(C_1)|$ is odd. By Theorem 9 there exists a 1-perfect forest F_1 in C_1 , such that p_1 (see Definition A) is the vertex of even degree in F_1 . Let $G' = G - V(C_1)$ and note that G' is connected and of even order. Therefore, by Theorem 1, there exists a 0-perfect forest, F' , in G' .

If $d_{F_1}(p_1) > 0$ then $F_1 \cup F'$ is a proper 1-perfect forest in G . Now consider the case when $d_{F_1}(p_1) = 0$. As $N(p_1) \cap V(G') = \{p_2\}$ (as $p_1p_2p_3$ is an induced path in G) we note that adding the edge p_1p_2 to $F_1 \cup F'$ gives us a proper 1-perfect forest in G (where p_2 is the only vertex of even degree). Thus, in the rest of the proof, we may assume that $|V(C_1)|$ is even.

Let $G' = G[V(C_1) \cup \{p_2, p_3\}]$ and note that G is connected and of even order. Furthermore $G' - \{p_2, p_3\}$ is connected (as $G' - \{p_2, p_3\} = C_1$). As p_1 is adjacent to p_2 but not to p_3 we note that $N_{G'}[p_2] \neq N_{G'}[p_3]$. By Lemma 4 we must therefore have $G' - p_2 \notin \mathcal{B}$ or $G' - p_3 \notin \mathcal{B}$. Let $i \in \{2, 3\}$ be chosen such that $G' - p_i \notin \mathcal{B}$, which by induction hypothesis implies that there is a proper 1-perfect forest F_1 in $G' - p_i$.

As G is 2-connected, we note that p_{5-i} is not a cut-vertex of G . Therefore every component in $G - \{p_2, p_3\}$ has an edge to p_i , which implies that $G - V(F_1)$ is connected and of even order (as both G and F_1 are of odd order). By Theorem 1 there exists a 0-perfect forest, F_2 , in $G - V(F_1)$. Now $F_1 \cup F_2$ is a proper 1-perfect forest in G . This completes the proof. ◀

A semiperfect forest F of G is called a *2-perfect forest* if exactly two vertices of F have even degree.

► **Lemma 3.** If G is a connected graph of odd order and $G \in \mathcal{B}$ then G does not contain a proper 1-perfect forest.

Proof. Let G be a connected graph of odd order and let $G \in \mathcal{B}$. We will prove that G contains no proper 1-perfect forest. We will prove this using induction on the number of blocks in G .

If G contains only one block then G is a complete graph of odd order. In this case, any forest where all trees are induced, can only contain trees of order 2 (and 1 if we allow isolated vertices). This implies that G cannot contain a proper 1-perfect forest as G has odd order. This completes the base case.

Now assume that G contains at least two blocks, which implies that G contains a cut-vertex, x . Let C_1, C_2, \dots, C_l be the components in $G - x$ and let $G_i = G[V(C_i) \cup \{x\}]$ for $i \in [l]$. For the sake of contradiction suppose that G contains a proper 1-perfect forest F and let F_i denote F restricted to G_i for $i \in [l]$. As F is a proper 1-perfect forest we note that $d_F(x) \geq 1$. Without loss of generality, assume that $d_{F_1}(x) \geq 1$. This implies that F_1 is a proper i -forest in G_1 where $i \in \{0, 1, 2\}$. However as $|V(G_1)|$ is odd (as $G \in \mathcal{B}$) this implies that F_1 is a proper 1-perfect forest in G_1 . This is a contradiction to $G_1 \in \mathcal{B}$ (as $G \in \mathcal{B}$). ◀

References

- 1 Jørgen Bang-Jensen, Eduard Eiben, Gregory Z. Gutin, Magnus Wahlström, and Anders Yeo. Component order connectivity in directed graphs. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14–18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPICs*, pages 2:1–2:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.
- 2 Daniel Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85–92, 1991. doi:10.1016/0012-365X(91)90098-M.
- 3 Yair Caro, Josef Lauri, and Christina Zarb. Two short proofs of the perfect forest theorem. *Theory and Applications of Graphs*, 4(1), 2017. article 4.
- 4 Robert Crowston, Mark Jones, and Matthias Mnich. Max-cut parameterized above the Edwards-Erdős bound. *Algorithmica*, 72(3):734–757, 2015.
- 5 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- 6 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.
- 7 Gregory Gutin and Anders Yeo. Lower bounds for Maximum Weighted Cut, 2021. arXiv: 2104.05536.
- 8 Gregory Gutin and Anders Yeo. Perfect forests in graphs and their extensions, 2021. arXiv: 2105.00254v1.
- 9 Gregory Z. Gutin. Note on perfect forests. *J. Graph Theory*, 82(3):233–235, 2016.
- 10 Gregory Z. Gutin and Anders Yeo. Constraint satisfaction problems parameterized above or below tight bounds: A survey. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond – Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 257–286. Springer, 2012.
- 11 Gregory Z. Gutin and Anders Yeo. Note on perfect forests in digraphs. *J. Graph Theory*, 85(2):372–377, 2017.
- 12 John E. Hopcroft and Robert Endre Tarjan. Efficient algorithms for graph manipulation [H] (algorithm 447). *Communications of ACM*, 16(6):372–378, 1973.
- 13 Daniel Lokshantov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15:1–15:31, 2014.
- 14 Daniel Lokshantov, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Balanced judicious bipartition is fixed-parameter tractable. *SIAM J. Discrete Mathematics*, 33(4):1878–1911, 2019.
- 15 László Lovász and Michael D. Plummer. *Matching Theory*. Akadémiai Kiadó, 1986.
- 16 Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1–3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978.
- 17 Alex D. Scott. On induced subgraphs with all degrees odd. *Graphs & Combinatorics*, 17(3):539–553, 2001.
- 18 Roded Sharan and Avi Wigderson. A new NC algorithm for perfect matching in bipartite cubic graphs. In *Fourth Israel Symposium on Theory of Computing and Systems, ISTCS 1996, Jerusalem, Israel, June 10–12, 1996, Proceedings*, pages 202–207. IEEE Computer Society, 1996.

On Deciding Linear Arithmetic Constraints Over p -adic Integers for All Primes

Christoph Haase  

Department of Computer Science, University of Oxford, UK

Alessio Mansutti  

Department of Computer Science, University of Oxford, UK

Abstract

Given an existential formula Φ of linear arithmetic over p -adic integers together with valuation constraints, we study the p -universality problem which consists of deciding whether Φ is satisfiable for all primes p , and the analogous problem for the closely related existential theory of Büchi arithmetic. Our main result is a coNEXP upper bound for both problems, together with a matching lower bound for existential Büchi arithmetic. On a technical level, our results are obtained from analysing properties of a certain class of p -automata, finite-state automata whose languages encode sets of tuples of natural numbers.

2012 ACM Subject Classification Theory of computation \rightarrow Logic

Keywords and phrases linear arithmetic, Büchi arithmetic, p -adic numbers, automatic structures

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.55

Funding This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 852769, ARIAT).



1 Introduction

In the light of the undecidability of Hilbert's tenth problem, the decidability of the Diophantine problem for addition and divisibility established by Lipshitz [20] is a non-trivial and interesting result. The latter problem consists of deciding whether a system of divisibility constraints of the form $p(\mathbf{x}) \mid q(\mathbf{x})$, with p and q being linear polynomials, has a solution over the integers. Lipshitz' proof of decidability relies on a local-to-global principle. He showed that every such system can be transformed into an equi-satisfiable one that has a solution if and only if an associated restricted system of linear equations with simple p -adic valuation constraints is satisfiable over the p -adic integers for every prime p . We call the latter problem the *p -universality problem*. To decide p -universality for the restricted class he considered, Lipshitz showed that it suffices to only check satisfiability for all primes p up to a certain threshold that can be computed from the input. One main result of this paper is to show that the latter result can be generalised: p -universality is decidable in coNEXP for *arbitrary* systems of linear equations over p -adic integers together with *general* linear p -adic valuation constraints. For linear equations with first-order variables ranging over the whole field of the p -adic numbers and restricted valuation constraints that allow to impose a partial order on the p -adic valuations of the first-order variables, a quantifier-elimination procedure was given by Dolzmann and Sturm from which it is possible to derive a coNEXP upper bound for p -universality in this setting [8]. Their result also shows that, in their setting, the set of those primes for which a solution exists is either finite or co-finite.

Linear arithmetic over p -adic integers with valuation constraints is closely related to *Büchi arithmetic*. Büchi arithmetic of base $p \geq 2$, p not necessarily prime, is the first-order theory of the structure $(\mathbb{N}, +, =, V_p)$, an extension of Presburger arithmetic with a unary V_p function



© Christoph Haase and Alessio Mansutti;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 55; pp. 55:1–55:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

such that $V_p(a) = b$ if and only if b is the largest power of p dividing a without remainder, i.e., there is some $k \in \mathbb{N}$ such that $b = p^k$, $b \mid a$ and $p \cdot b \nmid a$. Büchi showed that this theory is decidable using an automata-based approach, and conversely that Büchi arithmetic of base p defines the sets of numbers recognisable by p -automata, finite-state automata defining tuples of natural numbers encoded as words of tuples over the alphabet $\{0, \dots, p-1\}$ [5], though the latter result was incorrectly stated by Büchi and later correctly stated and proved by Bruyère [3]. One central line of research in Büchi arithmetic has been to understand the properties of this theory when the base p is variable. For instance, the celebrated Cobham-Semënov theorem states that if a set $M \subseteq \mathbb{N}^d$ is separately definable in Büchi arithmetic of multiplicatively independent bases p and q , then M is definable in Presburger arithmetic [7, 26]. Another main result of this paper is to show coNEXP-completeness of the analogue of p -universality for existential Büchi arithmetic: given an existential formula Ψ of Büchi arithmetic, decide whether Ψ is satisfiable in all bases $p \geq 2$. Note that p -universality does not imply definability in Presburger arithmetic as, for instance, the formula $V_p(x) = y$ is not definable in Presburger arithmetic, but it is p -universal.

Both coNEXP upper bounds are obtained by establishing doubly-exponential upper bounds on the smallest p for which a given formula becomes unsatisfiable. As a structural result, we obtain that for linear equations over p -adic integers with valuation constraints, the set of those primes p for which a given instance is satisfiable is precisely contained in an ultimately periodic set. On a technical level, our results are obtained by analysing properties of p -automata. While the latter have been studied for decades, only recently have they been instrumental in obtaining tight complexity bounds for long-standing open problems about the complexity of the satisfiability problem of the existential theories of the two arithmetic theories we consider in this paper [11]. A key observation we exploit for our approach is that the set of states of a p -automaton accepting the solutions of a system of linear Diophantine equations does *not* depend on p . Note that the quantifier-elimination approach employed by Dolzmann and Sturm [8] does not seem applicable in our setting as it works over the whole p -adic numbers and relies on them being a field. Moreover, Büchi arithmetic does not have a quantifier-elimination procedure, even when extended with additional predicates definable in existential Büchi arithmetic [13].

2 Preliminaries and main results

The symbols \mathbb{Z} , \mathbb{N} and \mathbb{Q} denote the set of integers, natural and rational numbers, respectively. We write \mathbb{P} for the set of prime numbers, and $\overline{\mathbb{Z}}$ to denote the set of integers extended with the symbol ∞ such that $n \leq \infty$ for all $n \in \mathbb{Z}$. All numbers are assumed to be encoded in binary, unless otherwise stated. For any object, we denote by $\langle \cdot \rangle$ the size of its encoding.

Linear arithmetic constraints over p -adic integers. Let $p \geq 2$ be a fixed prime number. Given a non-zero rational number $q \in \mathbb{Q}$, the p -adic valuation $v_p(q)$ is defined as the unique integer $k \in \mathbb{Z}$ such that $q = p^k \cdot \frac{a}{b}$ for $a, b \in \mathbb{Z}$ not divisible by p , and $v_p(0) = \infty$. The valuation v_p induces the p -adic absolute value $|\cdot|_p$ defined as $|q|_p = p^{-v_p(q)}$. The field of p -adic numbers \mathbb{Q}_p is obtained as the Cauchy completion of the field of the rational numbers under $|\cdot|_p$. Any p -adic number different from 0 has a unique p -adic expansion as an infinite power series $\sum_{i=k}^{\infty} a_i p^i$ for some $k \in \mathbb{Z}$, $a_k \neq 0$ and $a_i \in [0, p-1]$ for all $i \geq k$. The ring \mathbb{Z}_p of p -adic integers consists of all p -adic numbers for which this k is non-negative. By *linear arithmetic constraints over p -adic integers*, we refer to the first-order theory of the two-sorted structure $(\{\mathbb{Z}_p, \overline{\mathbb{Z}}\}, 0, 1, +, =, <, v_p)$. All constants, relational and functional symbols have their natural

semantics, and v_p is the p -adic valuation mapping p -adic integers to the valuation ring $\overline{\mathbb{Z}}$. For simplicity, we view the constants 0 and 1 as well as binary addition $+$ as being defined for both sorts. However, addition is restricted between elements of the *same* sort. The equality relation $=$ is defined on both $\overline{\mathbb{Z}}$ and \mathbb{Z}_p , whereas the less-than relation $<$ is restricted to the valuation ring $\overline{\mathbb{Z}}$. Usually, the letters u, v refer to first-order variables interpreted over \mathbb{Z}_p , and x, y, z refer to variables over $\overline{\mathbb{Z}}$. We rely on the axiom system for integer arithmetic enriched with infinity presented in [17] to treat linear terms over the valuation ring containing the symbol ∞ .

Note that we allow arbitrary Boolean combinations of linear inequalities to constraint valuations of the variables from \mathbb{Z}_p , whereas Dolzmann and Sturm [8] as well as Lipshitz [20] only allow restricted constraints of the form $v_p(u) \leq v_p(v)$.

Büchi arithmetic. Let $p \geq 2$ be a fixed integer. *Büchi arithmetic* of base p is the first-order theory of the structure $(\mathbb{N}, 0, 1, +, =, V_p)$, where the constants 0 and 1 and the relations $+$ and $=$ are interpreted in their natural semantics, and V_p is the unary function mapping every non-zero integer x to the largest power of p that divides x without remainder as defined in the introduction. For the purpose of this paper, as in [4] we define $V_p(0) = 1$, though other definitions such $V_p(0) = \infty$ are possible, but they do not change the sets of numbers definable in Büchi arithmetic. The decidability of Büchi arithmetic rests on the fact that Büchi arithmetic is an automatic structure in the sense of [14, 16, 2]. While full Büchi arithmetic is TOWER-complete [25], its existential fragment is only NP-complete [11].

Main decision problems and results. Both Büchi arithmetic and linear arithmetic constraints over p -adic integers are defined with respect to a fixed base $p \in \mathbb{N}$. In this paper, we treat p as a parameter, and, for a given formula Φ of *existential* Büchi arithmetic or *existential* linear arithmetic constraints over p -adic integers mentioning p , are interested in the following two decision problems:

- p -EXISTENCE: Is Φ satisfiable for *some* $p \geq 2$?
- p -UNIVERSALITY: Is Φ satisfiable for *every* $p \geq 2$?

When Φ is a formula of linear arithmetic constraints over p -adic integers, p above is additionally restricted to be a prime number. For the complexity of those decision problems, we stipulate that the V_p and v_p functions count as a single symbol in $\langle \Phi \rangle$ for any formula Φ . Note that p -universality and p -existence are not the complement of one and another: the formula $x \neq 2 \vee V_p(x) = 2$ of Büchi arithmetic has a solution for $p = 2$, but its negation is not p -universal. As the main results of this paper, we show:

► **Theorem 1.** *For both Büchi arithmetic and linear arithmetic constraints over p -adic integers, p -existence and p -universality are decidable in NEXP and coNEXP, respectively.*

► **Theorem 2.** *Deciding p -universality for Büchi arithmetic is coNEXP-hard.*

Further general notation. For an arbitrary set A , we write $\#A$ for its cardinality. If A is infinite, then $\#A = \infty$. For $a, b \in \mathbb{Z}$, we write $[a, b]$ for the set $\{a, a + 1, \dots, b\}$. Given a matrix $\mathbf{A} \in \mathbb{Z}^{n \times d}$ with components $a_{i,j} \in \mathbb{Z}$ ($i \in [1, n]$ and $j \in [1, d]$), the ∞ -norm of \mathbf{A} is defined as $\|\mathbf{A}\|_\infty \stackrel{\text{def}}{=} \max_{i=1, j=1}^{n, d} |a_{i,j}|$. We extend $\|\cdot\|_\infty$ to vectors in \mathbb{Z}^d by viewing them as elements of $\mathbb{Z}^{d \times 1}$. The $(1, \infty)$ -norm of \mathbf{A} is defined as $\|\mathbf{A}\|_{1, \infty} \stackrel{\text{def}}{=} \max_{i=1}^n \sum_{j=1}^d |a_{i,j}|$. Given a finite set $A \subseteq \mathbb{Z}^n$ of d integer vectors, we write $A^{\mathbf{M}}$ to denote the $n \times d$ matrix whose columns are the vectors in A , ordered following a lexicographic ordering. When clear from the context, we shall abbreviate $A^{\mathbf{M}}$ simply as \mathbf{A} . We write $\|A\|_\infty$ for $\|\mathbf{A}\|_\infty$.

Let $S: \mathbf{A} \cdot \mathbf{x} \geq \mathbf{c}$ be a system of linear inequalities with $\mathbf{A} \in \mathbb{Z}^{n \times d}$ and $\mathbf{c} \in \mathbb{Z}^n$. We write $\llbracket S \rrbracket$ for the solution set of S , that is the set of all $\mathbf{v} \in \mathbb{Z}^d$ such that $\mathbf{A} \cdot \mathbf{v} \geq \mathbf{c}$. We use $\llbracket S \rrbracket_{\geq 0}$ as a shorthand for $\llbracket S \rrbracket \cap \mathbb{N}^d$. Moreover, we define $\|S\| \stackrel{\text{def}}{=} \max(\|\mathbf{A}\|_\infty, \|\mathbf{c}\|_\infty)$. Finally, given a formula Φ of either Büchi arithmetic or linear arithmetic constraints over p -adic integers, we write $\|\Phi\|$ for the maximum absolute value of an integer appearing in Φ .

Deterministic p -automata and linear Diophantine equations. A central technical tool underlying the results of Theorem 1 are p -automata, a class of finite-state automata whose languages encode sets of natural numbers, see e.g. [4]. Given an integer $p \geq 2$, a p -automaton is a deterministic automaton over an alphabet $\Sigma_p^d := [0, p-1]^d$ for some positive integer d . A finite word $w = \mathbf{u}_k \cdots \mathbf{u}_0 \in (\Sigma_p^d)^*$ over Σ_p^d can be seen as encoding a d -tuple of non-negative integers in base p . We consider a *msd-first encoding* $\llbracket \cdot \rrbracket^*$, in which the most significant digit is on the left. Formally, $\llbracket w \rrbracket^* \in \mathbb{N}^d$ is defined as $\sum_{j=0}^k p^k \cdot \mathbf{u}_j$. Also note that for $w = \varepsilon$, the empty word, we have $\llbracket w \rrbracket^* = \mathbf{0}$.

Following [29], we define a p -automaton whose language is the msd-first encoding of all non-negative integer solutions of a system of linear equations.

► **Definition 3.** Let $S: \mathbf{A} \cdot \mathbf{x} = \mathbf{c}$ be a system of linear Diophantine equations with $\mathbf{A} \in \mathbb{Z}^{n \times d}$ and $\mathbf{c} \in \mathbb{Z}^n$. We define a p -automaton corresponding to S as $\mathcal{A}_p^*(S) \stackrel{\text{def}}{=} (Q, \Sigma_p^d, \delta, \mathbf{q}_0, F)$ with a set of states $Q = \mathbb{Z}^n$, transitions $\delta(\mathbf{q}, \mathbf{u}) = p \cdot \mathbf{q} + \mathbf{A} \cdot \mathbf{u}$ for all $\mathbf{q} \in Q$ and $\mathbf{u} \in \Sigma_p^d$, initial state $\mathbf{q}_0 = \mathbf{0}$, and final state $F = \{\mathbf{c}\}$.

For states $\mathbf{s}, \mathbf{t} \in Q$ and $\mathbf{u} \in \Sigma_p^d$, we write $\mathbf{s} \xrightarrow{\mathbf{u}}_{\mathbf{A}, p} \mathbf{t}$ whenever $\delta(\mathbf{s}, \mathbf{u}) = \mathbf{t}$. This notation is extended to words in the usual way: for a word $w \in (\Sigma_p^d)^*$, $\mathbf{s} \xrightarrow{w}_{\mathbf{A}, p} \mathbf{t}$ whenever there is $\mathbf{q} \in Q$ such that $\mathbf{s} \xrightarrow{w}_{\mathbf{A}, p} \mathbf{q} \xrightarrow{\mathbf{u}}_{\mathbf{A}, p} \mathbf{t}$. We write $\mathbf{s} \rightarrow_{\mathbf{A}, p} \mathbf{t}$ if $\mathbf{s} \xrightarrow{w}_{\mathbf{A}, p} \mathbf{t}$ holds for some $w \in (\Sigma_p^d)^*$, and omit the subscripts \mathbf{A} or p from $\rightarrow_{\mathbf{A}, p}$ when clear from the context.

As usual, under *regular* acceptance condition, a finite word $w \in (\Sigma_p^d)^*$ is accepted by the automaton $\mathcal{A} = \mathcal{A}_p^*(S)$ whenever $\mathbf{q}_0 \xrightarrow{w} \mathbf{f}$ for some $\mathbf{f} \in F$. The language $\mathcal{L}^*(\mathcal{A})$ of \mathcal{A} is the set of all words that are accepted by \mathcal{A} . Even though the automaton \mathcal{A} has infinitely many states, $\mathcal{L}^*(\mathcal{A})$ is a regular language since only finitely many *live states* can reach an accepting state.

► **Proposition 4** ([11], Prop. 5). Given the automaton $\mathcal{A}_p^*(S)$, only states $\mathbf{q} \in Q$ such that $\|\mathbf{q}\|_\infty \leq \max(\|\mathbf{A}\|_{1, \infty}, \|\mathbf{c}\|_\infty)$ can reach an accepting state.

Proposition 4 implies a bound on the cardinality of the set L of live states of the p -automaton $\mathcal{A}_p^*(S)$ as defined in Definition 3:

$$\#L \leq 2^n \cdot \max(\|\mathbf{A}\|_{1, \infty}, \|\mathbf{c}\|_\infty)^n \quad (1)$$

Observe that Proposition 4 also gives us a first key insight into deciding p -universality, as it shows that the set of live states of a p -automaton $\mathcal{A}_p^*(S)$ does *not* depend on the base p , but only on the system S . Deciding reachability in a p -automaton reduces to finding non-negative solutions to a certain system of Diophantine equations, as shown by the following proposition.

► **Proposition 5** ([11]). Given $\mathbf{s}, \mathbf{t} \in Q$, $k \in \mathbb{N}$ and $w \in (\Sigma_p^d)^k$, $\mathbf{s} \xrightarrow{w} \mathbf{t}$ iff $\mathbf{t} = p^k \cdot \mathbf{s} + \mathbf{A} \cdot \llbracket w \rrbracket^*$.

In view of the bounds on the set of live states given in (1), the length of the shortest word w witnessing $\mathbf{s} \rightarrow \mathbf{t}$ is exponential in $\langle S \rangle$. Of course, when $\mathbf{s} = \mathbf{0}$ and $\mathbf{t} = \mathbf{c}$, this bound is non-optimal, as von zur Gathen and Sieveking [27] have shown that any feasible system of

linear Diophantine equations S has a solution whose bit-size is polynomially bounded in $\langle S \rangle$. However, in the context of the p -universality problem, this bound on w is sufficient for us to establish the complexity upper bounds given by Theorem 1.

ω -regular acceptance condition and systems of equations over p -adic integers. A similar connection as in the previous paragraph can be established for systems of equations over p -adic integers [11]. In this setting, we consider infinite words $w = \mathbf{u}_0\mathbf{u}_1 \cdots \in (\Sigma_p^d)^\omega$ over Σ_p^d and view them as *lsd-first encodings* of d -tuples of p -adic integers in which the least significant digit is on the left. Formally, we define $\llbracket w \rrbracket^\omega \in \mathbb{Z}_p^d$ as $\sum_{j=0}^{\infty} p^j \cdot \mathbf{u}_j$.

Let $S: \mathbf{A} \cdot \mathbf{x} = \mathbf{c}$ be a system of linear equations with $\mathbf{A} \in \mathbb{Z}^{n \times d}$ and $\mathbf{c} \in \mathbb{Z}^n$, and let $w = \mathbf{u}_0\mathbf{u}_1 \cdots \in (\Sigma_p^d)^\omega$. We have $\mathbf{A} \cdot \llbracket w \rrbracket^\omega = \mathbf{c}$ if and only if $\mathbf{A} \cdot \llbracket w \rrbracket^\omega = \mathbf{c} \pmod{p^k}$ for all $k \in \mathbb{N}$. It follows, and was also discussed in [11], that $\mathbf{A} \cdot \llbracket w \rrbracket^\omega = \mathbf{c}$ if and only if for every $k \in \mathbb{N}$ there is $\mathbf{r} \in \mathbb{Z}^n$ such that $\mathbf{A} \cdot \llbracket \mathbf{u}_{k-1}\mathbf{u}_{k-2} \cdots \mathbf{u}_0 \rrbracket^* + \mathbf{r} \cdot p^k = \mathbf{c}$. By Proposition 5, the right hand side of this double implication expresses that the state \mathbf{r} can reach \mathbf{c} in the p -automaton $\mathcal{A}_p^*(S)$ by reading the word $\mathbf{u}_{k-1}\mathbf{u}_{k-2} \cdots \mathbf{u}_0$. So, $\mathbf{A} \cdot \llbracket w \rrbracket^\omega = \mathbf{c}$ is satisfied whenever the Büchi automaton obtained from $\mathcal{A}_p^*(S)$ by reversing every transition and making all states accepting has a non-empty language for the initial state \mathbf{c} . This ω -regular acceptance condition can equivalently be formulated as follows:

► **Proposition 6.** *For all $w = \mathbf{u}_0\mathbf{u}_1 \cdots \in (\Sigma_p^d)^\omega$, $\mathbf{A} \cdot \llbracket w \rrbracket^\omega = \mathbf{c}$ iff there is $\mathbf{r} \in \mathbb{Z}^n$ and a strictly ascending sequence $(\lambda_i)_{i \in \mathbb{N}}$ such that $\mathbf{r} \xrightarrow{\mathbf{u}_{\lambda_0-1} \cdots \mathbf{u}_{\lambda_0}}_{\mathbf{A}, p} \mathbf{c}$ and $\mathbf{r} \xrightarrow{\mathbf{u}_{\lambda_{j+1}} \cdots \mathbf{u}_{\lambda_j}}_{\mathbf{A}, p} \mathbf{r}$ for all $j \in \mathbb{N}$.*

Semi-linear set and ultimately periodic sets. Together with p -automata, to prove Theorem 1 we rely on well-known connections between solutions of systems of linear Diophantine equations and semi-linear sets. For $\mathbf{b} \in \mathbb{Z}^d$ and a finite set $P \subseteq \mathbb{Z}^d$ consisting of n elements, $L(\mathbf{b}, P)$ defines the *linear set* $\{\mathbf{x} \in \mathbb{Z}^d : \mathbf{x} = \mathbf{b} + \mathbf{P} \cdot \boldsymbol{\lambda} \text{ for some } \boldsymbol{\lambda} \in \mathbb{N}^n\}$. For a finite set $B \subseteq \mathbb{Z}^d$, $L(B, P)$ defines the *hybrid-linear set* $\bigcup_{\mathbf{b} \in B} L(\mathbf{b}, P)$. A *semi-linear set* is a finite union of hybrid-linear sets.

We use the following bound on the magnitude of the *bases* B and *periods* P of the set of solutions of a system of linear Diophantine equations, which is derived from [23].

► **Proposition 7** ([6], Prop. 4). *Let $\mathbf{A} \in \mathbb{Z}^{n \times d}$, $\mathbf{c} \in \mathbb{Z}^n$ and $S: \mathbf{A} \cdot \mathbf{x} = \mathbf{c}$. Then $\llbracket S \rrbracket_{\geq 0} = L(B, P)$ where $\|B\|_\infty \leq ((d+1) \cdot \|\mathbf{A}\|_\infty + \|\mathbf{c}\|_\infty + 1)^n$ and $\|P\|_\infty \leq (d \cdot \|\mathbf{A}\|_\infty + 1)^n$.*

Eventually, deciding p -existence and p -universality reduces to characterising the set of bases p for which an existential formula of Büchi arithmetic (or linear arithmetic constraints over p -adic integers) is satisfiable. This leads us to consider semi-linear sets in \mathbb{N} , which are equivalent to *ultimately periodic sets*, i.e., sets of definable as $F \cup L(T, q)$, where $q \in \mathbb{N}$ is the *period* of the ultimately periodic set, $F \subseteq \mathbb{N}$ is a finite set such that $\max F < \min T$, and $T \subseteq [t, t+q-1]$, where $t \in \mathbb{N}$ is the *threshold* of the ultimately periodic set.

Following [28], the essential building block leading to this change of representation, from one-dimensional semi-linear sets to ultimately periodic sets, is given by the proposition below.

► **Proposition 8.** *Let $M = L(B, P) \subseteq \mathbb{N}$. Then M is an ultimately periodic set with period $\text{gcd } P$ and threshold bounded by $\|B\|_\infty + \|P\|_\infty^2$.*

We recall bounds on union, intersection and set difference of ultimately periodic sets.

► **Proposition 9.** *Let M and N be two ultimately periodic sets with periods and thresholds respectively (p_1, t_1) and (p_2, t_2) . Then, $M \cup N$, $M \cap N$ and $M \setminus N$ are ultimately periodic sets with period $\text{lcm}(p_1, p_2)$ and threshold $\max(t_1, t_2)$.*

Since $\mathbb{N} = L(0, 1)$, Proposition 9 shows that the complement $\mathbb{N} \setminus M$ of an ultimately periodic set M is itself ultimately periodic, and has the same period and threshold as M .

For linear arithmetic constraints over p -adic integers, p -existence and p -universality restrict p to be prime numbers. We handle this restriction by using a variant of Linnik's theorem [19] to guarantee the existence of small primes on arithmetic progressions.

► **Proposition 10.** *There is a constant $c > 0$ such that for all co-prime $b, q \in \mathbb{N}$ there is some $p \in L(b, q) \cap \mathbb{P}$ such that $p \leq c \cdot (b \cdot r)^5$.*

Proof. Under the assumption that $b \in [1, q - 1]$, Linnik's theorem states that $L(b, q)$ contains a prime in $[1, d \cdot p^L]$ for fixed $d > 0$ and $L \in \mathbb{N}$. The best known bound for L is 5, as shown by Xylouris in [30]. To get rid of this additional restriction on b , consider a prime $s \in [b+1, 2(b+1)]$, whose existence follows from Bertrand's postulate [22]. From the primality of $s > b$ and the co-primality of b and q , we derive $\gcd(b, s \cdot q) = 1$ and $b < s \cdot r$. We can now safely apply Linnik's theorem, and derive that $L(b, s \cdot r) \subseteq L(b, q)$ contains a prime bounded by $c \cdot (b \cdot r)^5$ for some constant $c > 0$. ◀

Observe that every element of $L(b, q)$ is by definition divided by $\gcd(b, q)$. Hence, in the case where b and q are not co-prime, the only possible prime number appearing in $L(b, q)$ is b .

3 Exponential witnesses for p -existence and p -universality

For an existential formula Φ of either Büchi arithmetic or linear arithmetic constraints over p -adic integers, parametric in their base p , we write $\mathcal{B}(\Phi)$ for the set of bases $p \geq 2$ for which Φ is satisfiable. Note that, in defining $\mathcal{B}(\Phi)$ for linear arithmetic constraints over p -adic integers, we temporarily lift the primality condition on p . In this section, we establish the following result, which represents a crucial step in showing that the p -existence and p -universality problems are decidable in NEXP and coNEXP, respectively, proven in Section 4.

► **Theorem 11.** *Let Φ be an existential formula from Büchi arithmetic (resp. from linear arithmetic constraints over p -adic integers).*

- *If it exists, the smallest base $p \geq 2$ (resp. p prime) in $\mathcal{B}(\Phi)$ is bounded by $2^{2^{\mathcal{O}((\Phi)^2)}}$,*
- *If it exists, the smallest base $p \geq 2$ (resp. p prime) not in $\mathcal{B}(\Phi)$ is bounded by $2^{2^{\mathcal{O}((\Phi)^2)}}$.*

Whereas members of $\mathcal{B}(\Phi)$ are certificates of p -existence, a certificate for the non-universality of $\mathcal{B}(\Phi)$ can be retrieved from the “bases complement” $\overline{\mathcal{B}(\Phi)} \stackrel{\text{def}}{=} \mathbb{N} \setminus (\mathcal{B}(\Phi) \cup \{0, 1\})$. Consequently, a proof of Theorem 11 follows as soon as we show the following proposition.

► **Proposition 12.** *$\mathcal{B}(\Phi)$ is an ultimately periodic set with period and threshold in $2^{2^{\mathcal{O}((\Phi)^2)}}$.*

By Proposition 9, this result implies that $\overline{\mathcal{B}(\Phi)}$ is an ultimately periodic set with the same period and threshold as $\mathcal{B}(\Phi)$. Notice that for linear arithmetic constraints over p -adic integers the primality of the certificates can be obtained by an application of Linnik's theorem: consider the ultimately periodic representation $F \cup L(T, q)$ of $\mathcal{B}(\Phi)$, and suppose that it contains a prime. If $F \cup T$ contains a prime, then it is bounded by $2^{2^{\mathcal{O}((\Phi)^2)}}$. Otherwise, there is some $t \in T$ such that $L(t, q)$ contains a prime. So, t and q are co-prime (as $t \notin \mathbb{P}$), and by Linnik's theorem $L(t, q)$ has a prime bounded by $2^{2^{\mathcal{O}((\Phi)^2)}}$. Analogously, if $\mathcal{B}(\Phi)$ avoids a prime, then $\overline{\mathcal{B}(\Phi)}$ has a prime in $2^{2^{\mathcal{O}((\Phi)^2)}}$.

Proof of Proposition 12: Büchi arithmetic. Let Φ be a formula of existential Büchi arithmetic with parametric base p . To show Proposition 12, we introduce an abstraction of p -automata that we call *support graphs*. Support graphs are graphs that, while being independent from the base p , may correspond to paths of a p -automaton for a linear system $S: \mathbf{A} \cdot \mathbf{x} = \mathbf{c}$, and integrate auxiliary systems of inequalities that we use to enforce the satisfaction of formulae of the form $V_p(x) = y$, again independently of the choice of p .

► **Definition 13.** Let $n \in \mathbb{N}$, and consider a tuple of variables \mathbf{x} . A support graph on (n, \mathbf{x}) is a finite directed graph (V, E) with vertices $V \subseteq \mathbb{Z}^n$ and edges E of the form $\mathbf{s} \rightarrow_T \mathbf{t}$, where $\mathbf{s}, \mathbf{t} \in V$ and T is a system of linear inequalities with variables from \mathbf{x} .

A support graph can have multiple edges over the same two vertices, labelled with different systems of linear inequalities. We evaluate a support graph to the set of bases p for which it can be embedded into a p -automaton. Given $\mathbf{s}, \mathbf{t} \in V$ and a matrix $\mathbf{A} \in \mathbb{Z}^{n \times d}$, we define

$$\llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}} \stackrel{\text{def}}{=} \{z \in \mathbb{N} : \mathbf{t} = \mathbf{s} \cdot z + \mathbf{A} \cdot \mathbf{x}, z \geq 2 \text{ and } \|\mathbf{x}\|_{\infty} < z, \text{ for some } \mathbf{x} \in \llbracket T \rrbracket_{\geq 0}\}.$$

Notice that $\llbracket \mathbf{s} \rightarrow_{\top} \mathbf{t} \rrbracket_{\mathbf{A}}$, where \top is a (trivial) system of inequalities such that $\llbracket \top \rrbracket_{\geq 0} = \mathbb{N}^d$, corresponds to the set of bases $p \geq 2$ for which the p -automaton $\mathcal{A}_p^*(S)$ has a one-step transition from \mathbf{s} to \mathbf{t} . As we only look at non-negative values for z and \mathbf{x} , we can introduce slack variables to translate the inequalities $z \geq 2$, $\|\mathbf{x}\|_{\infty} < z$, as well as all the ones in T , into equalities. This allows us to apply Proposition 7, followed by Proposition 8, to characterise $\llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}}$ as an ultimately periodic set. Below, let $\|\mathbf{s} \rightarrow_T \mathbf{t}\|_{\infty} \stackrel{\text{def}}{=} \max(\|\mathbf{s}\|_{\infty}, \|\mathbf{t}\|_{\infty}, \|T\|)$.

► **Lemma 14.** Let $\mathbf{A} \in \mathbb{Z}^{n \times d}$, $\mathbf{s}, \mathbf{t} \in \mathbb{Z}^n$ and let T be a linear system of m inequalities. The set $\llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}}$ is an ultimately periodic set with period and threshold bounded by $U^{\mathcal{O}(k \log k)}$, where $k = n + d + m$ and $U = \max(2, \|\mathbf{A}\|_{\infty}, \|\mathbf{s} \rightarrow_T \mathbf{t}\|_{\infty})$.

Given a support graph \mathcal{G} with edges e_1, \dots, e_{ℓ} , we write $\llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$ for $\bigcap_{i \in [1, \ell]} \llbracket e_i \rrbracket_{\mathbf{A}}$, i.e. the set of $p \geq 2$ such that, for every edge $\mathbf{s} \rightarrow_T \mathbf{t}$ of \mathcal{G} , the transition $\mathbf{s} \xrightarrow{\mathbf{u}}_{p, \mathbf{A}} \mathbf{t}$ holds for some tuple $\mathbf{u} \in \Sigma_p^d$ satisfying T . By Proposition 9 and Lemma 14, $\llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$ is ultimately periodic.

To prove Proposition 12, we first translate the formula Φ (possibly by introducing slack variables to replace inequalities with equalities) in a disjunctive normal form with $2^{\mathcal{O}(\langle \Phi \rangle)}$ disjuncts have the form $\mathbf{A} \cdot \mathbf{x} = \mathbf{c} \wedge \bigwedge_{i \in I} V_p(x_i) = y_i$, where $\mathbf{A} \in \mathbb{Z}^{n \times d}$, $\mathbf{c} \in \mathbb{Z}^n$, and all variables are among the ones in \mathbf{x} . We further manipulate each of these disjuncts by considering all linear orderings among the variables y_i ($i \in I$). Variables that are set to be equal in an ordering can be substituted accordingly, so that Φ is found to be equivalent to a disjunction of $2^{\mathcal{O}(\langle \Phi \rangle \log \langle \Phi \rangle)}$ formulae of size $\mathcal{O}(\langle \Phi \rangle)$ that have the form

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{c} \wedge \bigwedge_{(i,j) \in J} V_p(x_i) = y_j \wedge \bigwedge_{j \in [1, m]} y_j < y_{j-1} \quad (2)$$

where $J \subseteq I \times [0, m]$ is a binary relation that is functional and surjective on its first component.

Let ψ be a formula of the form in (2). We aim at characterising $\mathcal{B}(\psi)$ as an ultimately periodic set. Recall that, by Proposition 5, solutions of the system $S: \mathbf{A} \cdot \mathbf{x} = \mathbf{c}$ are values $\llbracket w \rrbracket^* \in \mathbb{N}^d$ for some $w \in (\Sigma_p^d)^*$ such that $\mathbf{0} \xrightarrow{w}_{\mathbf{A}, p} \mathbf{c}$. Moreover, a constraint $V_p(x) = y$ restricts the variables x and y to be such that, in their base- p msd representation, $y \in \{0\}^{\ell} \cdot \{1\} \cdot \{0\}^r$ and $x \in [0, p-1]^{\ell} \cdot [1, p-1] \cdot \{0\}^r$, for some $\ell, r \in \mathbb{N}$. Consequently, in order for $\llbracket w \rrbracket^*$ to be a solution of (2), the word w must admit a decomposition $w_0 \cdot \mathbf{u}_0 \cdot w_1 \cdots w_m \cdot \mathbf{u}_m \cdot w_{m+1}$ such that $\mathbf{u}_0, \dots, \mathbf{u}_m \in \Sigma_p^d$,

$$\mathbf{0} = \mathbf{s}_0 \xrightarrow{w_0}_{\mathbf{A}, p} \mathbf{t}_0 \xrightarrow{\mathbf{u}_0}_{\mathbf{A}, p} \mathbf{s}_1 \cdots \mathbf{s}_m \xrightarrow{w_m}_{\mathbf{A}, p} \mathbf{t}_m \xrightarrow{\mathbf{u}_m}_{\mathbf{A}, p} \mathbf{s}_{m+1} \xrightarrow{w_{m+1}}_{\mathbf{A}, p} \mathbf{t}_{m+1} = \mathbf{c}, \quad (3)$$

where $\mathbf{s}_1, \dots, \mathbf{s}_{m+1}, \mathbf{t}_0, \mathbf{t}_m$ are (intermediate) live states, and every \mathbf{u}_j and w_j with $j \in [0, m]$ shall satisfy the constraints induced by the function V_p together with the ordering on the variables y_j . In particular, following the aforementioned decomposition for the variables x and y appearing in a constraint $V_p(x) = y$, for every $j \in [0, m]$ the values of \mathbf{u}_j for the variables $x_1, \dots, x_{\#I}$ and y_1, \dots, y_m shall satisfy the system U_j :

$$\begin{cases} y_j = 1, & x_i \geq 1 \quad : i \in I \text{ and } V_p(x_i) = y_j \text{ occurs in (2),} \\ y_k = 0 \quad : k \in [1, m] \setminus \{j\}, & x_i = 0 \quad : i \in I \text{ and } V_p(x_i) = y_k \text{ occurs in (2) for some } k < j. \end{cases}$$

whereas at each position of the word w_j ($j \in [0, m+1]$) shall satisfy the system W_j :

$$\begin{cases} y_k = 0 \quad : k \in [1, m], & x_i = 0 \quad : i \in I \text{ and } V_p(x_i) = y_k \text{ occurs in (2), for some } k < j. \end{cases}$$

Hence, paths as in (3) can be abstracted into support graphs with vertices from the set of live states of $\mathcal{A}_p^*(S)$ and having the form

$$\mathbf{0} = \mathbf{s}_0 \xrightarrow{W_0^j} \mathbf{t}_0 \xrightarrow{U_0} \mathbf{s}_1 \dots \mathbf{s}_m \xrightarrow{W_m^j} \mathbf{t}_m \xrightarrow{U_m} \mathbf{s}_{m+1} \xrightarrow{W_{m+1}^{j_{m+1}}} \mathbf{t}_{m+1} = \mathbf{c}, \quad (4)$$

where $\mathbf{s} \xrightarrow{j} \mathbf{t}$ is short for a path of length j going from \mathbf{s} to \mathbf{t} , and with arrows labelled by the system of inequalities T , and for every $i \in [0, m+1]$, j_i is the length of w_i .

► **Lemma 15.** *Let ψ be a formula as in (2).*

- *For every support graph \mathcal{G} of the form described in (4), $[\mathcal{G}]_{\mathbf{A}} \subseteq \mathcal{B}(\psi)$.*
- *For every $p \in \mathcal{B}(\psi)$ there is a support graph \mathcal{G} as in (4) such that $p \in [\mathcal{G}]_{\mathbf{A}}$.*

Proof. Let \mathbb{G} be the set of support graphs of the form in (4). The lemma equivalently states that $\mathcal{B}(\psi) = \bigcup_{\mathcal{G} \in \mathbb{G}} [\mathcal{G}]_{\mathbf{A}}$. Below, we refer to the first and second points in the lemma as the two inclusions \supseteq and \subseteq of this equality.

(\supseteq): Let \mathcal{G} be a support graph in \mathbb{G} , and consider $p \in [\mathcal{G}]_{\mathbf{A}}$. Notice that, by definition, this means that for every edge $\mathbf{s} \xrightarrow{T} \mathbf{t}$ of \mathcal{G} there is $\mathbf{u} \in \Sigma_p^d$ such that $\mathbf{s} \xrightarrow{\mathbf{u}}_{\mathbf{A}, p} \mathbf{t}$ and $\mathbf{u} \in [T]$. From (4), there is a path

$$\mathbf{0} = \mathbf{s}_0 \xrightarrow{w_0}_{\mathbf{A}, p} \mathbf{t}_0 \xrightarrow{\mathbf{u}_0}_{\mathbf{A}, p} \mathbf{s}_1 \dots \mathbf{s}_m \xrightarrow{w_m}_{\mathbf{A}, p} \mathbf{t}_m \xrightarrow{\mathbf{u}_m}_{\mathbf{A}, p} \mathbf{s}_{m+1} \xrightarrow{w_{m+1}}_{\mathbf{A}, p} \mathbf{t}_{m+1} = \mathbf{c},$$

where $w \stackrel{\text{def}}{=} w_0 \cdot \mathbf{u}_0 \cdot w_1 \cdot \dots \cdot w_m \cdot \mathbf{u}_m \cdot w_{m+1} \in (\Sigma_p^d)^*$, $\mathbf{u}_0, \dots, \mathbf{u}_m \in \Sigma_p^d$, every \mathbf{u}_j satisfies U_j and every symbol in w_j satisfies W_j . Clearly, $\mathbf{A} \cdot [w] = \mathbf{c}$. From the definition of the systems U_0, \dots, U_m and W_0, \dots, W_{m+1} , we obtain that in w the value for the variable y_j ($j \in [0, m]$) has a base- p msd representation of the form $\{0\}^{\ell_j} \cdot \{1\} \cdot \{0\}^{r_j}$, for some $\ell_j, r_j \in \mathbb{N}$ such that $\ell_j + 1 + r_j$ corresponds to the length of w . This means that every y_j is a power of p . Moreover, $r_{j-1} > r_j$ for every $j \in [1, m]$, and therefore $y_j < y_{j-1}$. Lastly, consider $(i, j) \in J$, so that $V_p(x_i) = y_j$ appears in ψ . The systems U_0, \dots, U_m and W_0, \dots, W_{m+1} force the base- p msd encoding of x_i to belong to the language $[0, p-1]^{\ell_j} \cdot [1, p-1] \cdot \{0\}^{r_j}$. We conclude that the formula $V_p(x_i) = y_j$ holds. So, ψ is satisfiable with respect to the base p , i.e., $p \in \mathcal{B}(\psi)$.

(\subseteq): Follows conversely to the other inclusion. Suppose ψ satisfiable with respect to the base p . Consider a word $w \in (\Sigma_p^d)^*$ such that $[w]^*$ is a solution of ψ . From $\bigwedge_{(i,j) \in J} V_p(x_i) = y_j$ we conclude that the base- p msd encodings of x_i and y_j belong to $\{0\}^{\ell_j} \cdot \{1\} \cdot \{0\}^{r_j}$ and $[0, p-1]^{\ell_j} \cdot [1, p-1] \cdot \{0\}^{r_j}$, respectively, for some ℓ_j and r_j such that $\ell_j + 1 + r_j$ corresponds to the length of w . From $y_j < y_{j-1}$ ($j \in [1, m]$), $r_{j-1} > r_j$. Hence, w admits a decomposition $w_0 \cdot \mathbf{u}_0 \cdot w_1 \cdot \dots \cdot w_m \cdot \mathbf{u}_m \cdot w_{m+1}$ such that $\mathbf{u}_0, \dots, \mathbf{u}_m \in \Sigma_p^d$,

$$\mathbf{0} = \mathbf{s}_0 \xrightarrow{w_0}_{\mathbf{A}, p} \mathbf{t}_0 \xrightarrow{\mathbf{u}_0}_{\mathbf{A}, p} \mathbf{s}_1 \dots \mathbf{s}_m \xrightarrow{w_m}_{\mathbf{A}, p} \mathbf{t}_m \xrightarrow{\mathbf{u}_m}_{\mathbf{A}, p} \mathbf{s}_{m+1} \xrightarrow{w_{m+1}}_{\mathbf{A}, p} \mathbf{t}_{m+1} = \mathbf{c},$$

where $s_1, \dots, s_{m+1}, t_0, t_m$ are live states. Moreover, every u_j is a solution of U_j and every symbol in the word w_j is a solution of W_j . Let \mathcal{G} be the support graph with edges $t_0 \rightarrow_{U_0} s_1, \dots, t_m \rightarrow_{U_j} s_{m+1}$ together with $i \rightarrow_{W_j} i'$, for every $j \in [0, m+1]$ and every two states i, i' such that $i \rightarrow_{\mathbf{A}, p} i'$ appears in the path going from s_j to t_j . The graph \mathcal{G} is of the form in (4), and $p \in \llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$. \blacktriangleleft

Only finitely many support graphs have the form described in (4), as they all have vertices from the finite set of live states of $\mathcal{A}_p^*(S)$, and edges with labels from a finite set of linear systems. So, Lemma 15 implies that the set $\mathcal{B}(\psi)$ is equivalent to a finite union of $\llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$, for which we can obtain an ultimately periodic representation according to Proposition 9 and Lemma 14.

► Lemma 16. *Let ψ be as in (2). Then $\mathcal{B}(\psi)$ is ultimately periodic with threshold in $U^{\mathcal{O}(k \log k)}$ and period in $U^{\mathcal{O}(\ell \cdot k \log k)}$, where $U = \max(2, \|\mathbf{A}\|_{1, \infty}, \|\mathbf{c}\|_{\infty})$, $k = n + 3d^2$ and $\ell = U^{4n}$.*

Proof. Let \mathbb{G} be the finite family of support graphs such that $\mathcal{B}(\psi) = \bigcup_{\mathcal{G} \in \mathbb{G}} \llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$, according to Lemma 15. Every edge $s \rightarrow_T t$ of a support graph $\mathcal{G} \in \mathbb{G}$ is such that $\|s\|_{\infty}, \|t\|_{\infty} \leq \max(\|\mathbf{A}\|_{1, \infty}, \|\mathbf{c}\|_{\infty})$ and T is a system among $U_0, \dots, U_m, W_0, \dots, W_{m+1}$. Hence, all the graphs in $\bigcup_{j \in J} \mathbb{G}_j$ are built from a set E of $(2 \cdot \max(\|\mathbf{A}\|_{1, \infty}, \|\mathbf{c}\|_{\infty}))^{2n} \cdot (2m+3) \leq \mathcal{O}(\ell)$ edges (note: $m \leq \#I \leq d^2$). Each possible linear system T labelling an edge in E has at most $2d^2$ inequalities, with coefficients and constants in $\{0, 1\}$. By Lemma 14, each edge $e \in E$ is such that $\llbracket e \rrbracket_{\mathbf{A}}$ is an ultimately periodic set with period and threshold bounded by $U^{\mathcal{O}(k \log k)}$. By Proposition 9, taking unions and intersections of sets $\llbracket e \rrbracket_{\mathbf{A}}$ with $e \in E$, as for instance $\mathcal{B}(\psi) = \bigcup_{\mathcal{G} \in \mathbb{G}} \llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$, always yields an ultimately periodic set with threshold in $U^{\mathcal{O}(k \log k)}$ and period in $U^{\mathcal{O}(\ell \cdot k \log k)}$. \blacktriangleleft

The bounds U , k and ℓ established in Lemma 16 for the threshold and the period of $\mathcal{B}(\varphi)$ can be restated in terms of the size of the initial formula Φ as follows: $U \leq 2^{\mathcal{O}(\langle \Phi \rangle)}$, $k \leq \mathcal{O}(\langle \Phi \rangle^2)$ and $\ell \leq 2^{\mathcal{O}(\langle \Phi \rangle^2)}$. This is sufficient to conclude that Proposition 12 holds. Indeed, the formula Φ is equivalent to a disjunction $\bigvee_{k \in K} \psi_k$ of formulae ψ_k of the form in (2), with $\#K \leq 2^{\mathcal{O}(\langle \Phi \rangle \log \langle \Phi \rangle)}$. This means that the set $\mathcal{B}(\Phi)$ is the union of all $\mathcal{B}(\psi_k)$ with $k \in K$. We apply Proposition 9 to obtain a representation of $\mathcal{B}(\Phi)$ as an ultimately periodic set with threshold bounded by $2^{\mathcal{O}(\langle \Phi \rangle^3 \log \langle \Phi \rangle)}$ and period bounded by $2^{2^{\mathcal{O}(\langle \Phi \rangle^2)}}$.

Proof of Proposition 12: Linear arithmetic constraints over p -adic integers. We now establish Proposition 12 for the case of Φ being an existential formula of linear arithmetic constraints over p -adic integers with parametric base p . For brevity, all proofs in this section are relegated to Appendix B. The crucial difference from the proof of Proposition 12 for Büchi arithmetic is that, differently from the V_p function, the p -adic valuation v_p induces constraints related to the relative lengths of subwords of the infinite words accepted by the p -automaton. For instance, to satisfy the formula $v_p(u) = 3 \cdot v_p(v) \wedge v_p(v) \geq 1$, the base- p lsd-first representation of u and v must obey the following constraints:

$$\begin{aligned} u &\in \{0\}^i \{0\} \quad \{0\}^{2i+1} [1, p-1] (\Sigma_p)^\omega, \\ v &\in \{0\}^i [1, p-1] (\Sigma_p)^{2i+1} \Sigma_p \quad (\Sigma_p)^\omega. \end{aligned}$$

where $i \geq 1$. In particular, we notice that the length of the maximal all-zeros prefix of v fixes the length of the maximal all-zeros prefix of u , and vice versa. This reflects in the proof of Proposition 12 where, instead of only considering support graphs that are linear structures in the sense of (4), we must consider arbitrary graphs and establish ultimately periodic representations of the lengths of their paths. To do so, we rely on the following

result on the lengths of words accepted by a *nondeterministic finite automaton* (NFA) over a *unary alphabet*. Recall that an NFA is a tuple $\mathcal{A} = (Q, \delta, I, F)$ where Q is a finite set of states, $\delta \subseteq Q \times Q$ is a transition relation, and $I, F \subseteq Q$ are set of initial and final states, respectively.

► **Proposition 17** ([24]). *Given an unary NFA $\mathcal{A} = (Q, \delta, I, F)$ with $s = \#Q$, one can construct in time $\mathcal{O}(s^2(s + \#\delta))$ a set $L = \bigcup_{k \in K} L(b_k, q_k)$ characterising the lengths of the words accepted by \mathcal{A} , with $\#K \leq \mathcal{O}(s^2)$, $b_k \leq (2 \cdot s + 1) \cdot s$ and $q_k \leq s$.*

Of course, other modifications with respect to the treatment of Büchi arithmetic are required: the support graphs must take into account the ω -regular acceptance condition defined in Proposition 6, and we also have to deal with the two-sorted structure of the theory.

Moving to the proof of Proposition 12, similarly to the case of Büchi arithmetic we start by manipulating the formula Φ and obtain a disjunctive normal form where each of the $2^{\mathcal{O}(\langle \Phi \rangle \log \langle \Phi \rangle)}$ disjuncts are of size $\mathcal{O}(\langle \Phi \rangle)$ and have the following form:

$$\mathbf{A} \cdot \mathbf{u} = \mathbf{c} \wedge \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d} \wedge \bigwedge_{(i,j) \in J} v_p(u_i) = x_j \wedge \bigwedge_{j \in [1,r]} x_{j-1} < x_j \quad (5)$$

where $\mathbf{A} \in \mathbb{Z}^{n \times d}$, $\mathbf{c} \in \mathbb{Z}^n$, $\mathbf{B} \in \mathbb{Z}^{m \times e}$, $\mathbf{d} \in \mathbb{Z}^m$ and $J \subseteq I \times [0, r]$ is a binary relation that is functional and surjective on its first component. Each u_i with $i \in I$ is a variable among \mathbf{u} interpreted over \mathbb{Z}_p , and each x_j with $j \in [0, r]$ is a variable among \mathbf{x} , interpreted over \mathbb{Z} . Notice that restricting the interpretation of \mathbf{x} from \mathbb{Z} to \mathbb{Z} is without loss of generality: when bringing the formula Φ in disjunctive normal form, we can introduce tautologies of the form $x < \infty \vee x = \infty$, for each of the variables x in \mathbf{x} . Then, following the axiom system presented in [17], disjuncts where $x = \infty$ holds can be easily modified so that x is eliminated.

According to Proposition 6, solutions of the system $S: \mathbf{A} \cdot \mathbf{u} = \mathbf{c}$ over the p -adic integers are values $\llbracket w \rrbracket^\omega \in \mathbb{Z}_p^d$ for some infinite word $w = \mathbf{u}_0 \mathbf{u}_1 \dots \in (\Sigma_p^d)^\omega$ such that there is a live state $\mathbf{r} \in \mathbb{Z}^n$ of the p -automaton $\mathcal{A}_p^*(S)$ and an infinite sequence $\lambda_0 < \lambda_1 < \dots$ for which $\mathbf{r} \xrightarrow{\mathbf{u}_{\lambda_0-1} \dots \mathbf{u}_{\lambda_0}}_{\mathbf{A}, p} \mathbf{c}$ and $\mathbf{r} \xrightarrow{\mathbf{u}_{\lambda_{j+1}} \dots \mathbf{u}_{\lambda_j}}_{\mathbf{A}, p} \mathbf{r}$ for all $j \in \mathbb{N}$. Moreover, a constraint $v_p(u) = x$ restricts the variables u and x to be such that, in the base p lsd-first representation of u , we have $u \in \{0\}^x \cdot [1, p-1] \cdot [0, p-1]^\omega$. Consequently, in order for $(\llbracket w \rrbracket^\omega, \mathbf{x})$ to be a solution of (5), in addition to $\mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$, the word w must have a prefix of the form $w_0 \cdot \mathbf{v}_0 \cdot w_1 \dots w_r \cdot \mathbf{v}_r \cdot w_{r+1}$ such that $\mathbf{v}_0, \dots, \mathbf{v}_r \in \Sigma_p^d$, the word $w_0 \in (\Sigma_p^d)^*$ has length x_0 , each $w_i \in (\Sigma_p^d)^*$ with $i \in [1, r]$ has length $x_i - (x_{i-1} + 1) \geq 0$, and

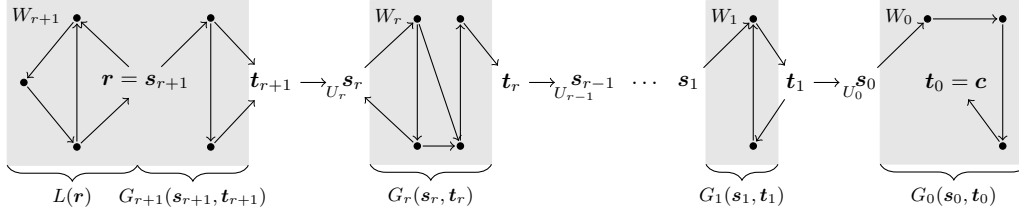
$$\mathbf{r} = \mathbf{s}_{r+1} \xrightarrow{(w_{r+1})^R}_{\mathbf{A}, p} \mathbf{t}_{r+1} \xrightarrow{\mathbf{v}_r}_{\mathbf{A}, p} \mathbf{s}_r \dots \mathbf{s}_1 \xrightarrow{(w_1)^R}_{\mathbf{A}, p} \mathbf{t}_1 \xrightarrow{\mathbf{v}_0}_{\mathbf{A}, p} \mathbf{s}_0 \xrightarrow{(w_0)^R}_{\mathbf{A}, p} \mathbf{t}_0 = \mathbf{c} \quad (6)$$

where each $(w_i)^R$ with $i \in [0, r+1]$ is the reverse of the word w_i , and \mathbf{r} is a live state of $\mathcal{A}_p^*(S)$ for which the ω -regular condition of Proposition 6 is satisfied. Following the decomposition for the variable u appearing in a constraint $v_p(u) = x$ given above, for every $j \in [0, r]$ the values of \mathbf{v}_j for the variables $u_1, \dots, u_{\#I}$ shall satisfy the system U_j :

$$\begin{cases} u_i \geq 1 & : i \in I \text{ and } v_p(u_i) = x_j \text{ occurs in (5),} \\ u_i = 0 & : i \in I \text{ and } v_p(u_i) = x_k \text{ occurs in (5), for some } k \in [j+1, r]. \end{cases}$$

whereas at each position of the word w_j ($j \in [0, r+1]$) shall satisfy the system W_j :

$$\begin{cases} u_i = 0 & : i \in I \text{ and } v_p(u_i) = x_k \text{ occurs in (5), for some } k \in [j, r]. \end{cases}$$



■ **Figure 1** A support graphs for existential linear arithmetic constraints over p -adic integers.

Given a formula ψ of the form in (5), we abstract paths in the p -automaton $\mathcal{A}_p^*(S)$ induced by infinite words such as the word w above, by introducing a family of support graphs, denoted by $\mathbb{G}(\psi)$. Each support graph $\mathcal{G} \in \mathbb{G}(\psi)$ has live states of $\mathcal{A}_p^*(S)$ as vertices, and its set of edges can be partitioned in the following sets, for some *intermediate live states* $\mathbf{r}, \mathbf{s}_0, \dots, \mathbf{s}_{r+1}, \mathbf{t}_0, \dots, \mathbf{t}_{r+1}$ such that $\mathbf{r} = \mathbf{s}_{r+1}$ and $\mathbf{t}_0 = \mathbf{c}$:

- $C(\mathbf{r})$: a set of edges of the form $\mathbf{s} \rightarrow_{W_{r+1}} \mathbf{t}$ that describes a connected graph with a non-empty path from \mathbf{r} to itself (as required by the ω -regular condition of Proposition 6),
- $G_j(\mathbf{s}_j, \mathbf{t}_j)$, with $j \in [0, r+1]$: a set of edges of the form $\mathbf{s} \rightarrow_{W_j} \mathbf{t}$ that describes a connected graph with a (possibly empty) path going from \mathbf{s}_j to \mathbf{t}_j ,
- $\{\mathbf{t}_{j+1} \rightarrow_{U_j} \mathbf{s}_j\}$, for every $j \in [0, r]$.

As the set of live states of $\mathcal{A}_p^*(S)$ and the set of all linear systems U_j and W_j considered are finite, so is the set $\mathbb{G}(\psi)$. Figure 1 depicts a support graph from $\mathbb{G}(\psi)$. We say that \mathcal{G} *generates the length values* $(v_0, \dots, v_r) \in \mathbb{N}^{r+1}$ if \mathcal{G} has a path of the form

$$\mathbf{t}_{r+1} \rightarrow_{U_r} \mathbf{s}_r \xrightarrow{v_r - (v_{r-1} + 1)} \mathbf{t}_r \rightarrow_{U_{r-1}} \mathbf{s}_{r-1} \dots \mathbf{s}_1 \xrightarrow{v_1 - (v_0 + 1)} \mathbf{t}_1 \rightarrow_{U_0} \mathbf{s}_0 \xrightarrow{v_0} \mathbf{t}_0 = \mathbf{c} \quad (7)$$

Exactly as in the case of Büchi arithmetic, we aim at characterising $\mathcal{B}(\psi)$ as a union over a subset B of $\{\llbracket \mathcal{G} \rrbracket_{\mathbf{A}} : \mathcal{G} \in \mathbb{G}(\psi)\}$. According to the definition of $\mathcal{G} \in \mathbb{G}(\psi)$, the set $\llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$ consists of some of the bases $p \geq 2$ for which, if we disregard the constraints imposed by the system of inequalities $\mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$, the formula ψ is satisfiable. To account for this system, we need to characterise the set of all length values that can be generated from \mathcal{G} , and check whether $\mathbf{B} \cdot \mathbf{x} \geq \mathbf{d} \wedge \bigwedge_{j \in [0, r]} x_j = v_j$ can be satisfied with respect to one of these length values (v_0, \dots, v_r) . This check, which we now formalise, does not depend on the base- p , so that either $\llbracket \mathcal{G} \rrbracket_{\mathbf{A}} \subseteq \mathcal{B}(\psi)$ or \mathcal{G} can be discarded when constructing the set B .

Consider $\mathcal{G} \in \mathbb{G}(\psi)$, with intermediate live states $\mathbf{s}_0, \dots, \mathbf{s}_{r+1} = \mathbf{r}, \mathbf{c} = \mathbf{t}_0, \dots, \mathbf{t}_{r+1}$. For every $j \in [0, r]$, we construct from the set of edges $G_j(\mathbf{s}_j, \mathbf{t}_j)$ the unary NFA (Q, δ, I, F) where Q is the set of live states appearing in some of the edges of $G_j(\mathbf{s}_j, \mathbf{t}_j)$, $I = \{\mathbf{s}_j\}$ and $F = \{\mathbf{t}_j\}$, and $\delta = \{(\mathbf{s}, \mathbf{t}) \in Q^2 : \mathbf{s} \rightarrow_{W_j} \mathbf{t} \in G_j(\mathbf{s}_j, \mathbf{t}_j)\}$. By Proposition 17, the set L_j of the lengths of the words accepted by this automaton is ultimately periodic. To obtain the length values that can be generated by \mathcal{G} , we combine the lengths of the sets L_0, \dots, L_r , and construct the following set $\oplus(L_0, \dots, L_r)$:

$$\oplus(L_0, \dots, L_r) \stackrel{\text{def}}{=} \left\{ \begin{pmatrix} \ell_0 \\ 1 + \ell_0 + \ell_1 \\ \dots \\ r + \sum_{i=0}^r \ell_i \end{pmatrix} \in \mathbb{N}^{r+1} : \ell_j \in L_j \text{ for all } j \in [0, r] \right\} \quad (8)$$

Below, we set $U \stackrel{\text{def}}{=} \max(2, \|\mathbf{A}\|_{1, \infty}, \|\mathbf{c}\|_{\infty})$, so that the live states of S are at most U^{2n} .

► **Lemma 18.** *The set $\oplus(L_0, \dots, L_r)$ contains all length values generated by \mathcal{G} . One can construct in time $\mathcal{O}(r^2) \cdot U^{\mathcal{O}(n \cdot r)}$ a representation of $\oplus(L_0, \dots, L_r)$ as a semi-linear set $\bigcup_{k \in K} L(\mathbf{b}_k, P_k)$, where $\#K \leq U^{\mathcal{O}(n \cdot r)}$, $\#P \leq r + 1$, $\|P\|_{\infty} \leq U^{2n}$ and $\|\mathbf{b}_k\|_{\infty} \leq \mathcal{O}(r \cdot U^{4n})$.*

As already stated, the set $L_{\mathcal{G}}$ allows us to characterise $\mathcal{B}(\psi)$ as a union over some of the sets in $\{\llbracket \mathcal{G} \rrbracket_{\mathbf{A}} : \mathcal{G} \in \mathbb{G}(\psi)\}$. This is formalised by the two following lemma, analogous to Lemma 15 established for Büchi arithmetic. For brevity, we write $\mathcal{G} \vdash \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$ whenever there is a length value $(v_0, \dots, v_r) \in L_{\mathcal{G}}$ such that $\mathbf{B} \cdot \mathbf{x} \geq \mathbf{d} \wedge \bigwedge_{j \in [0, r]} x_j = v_j$ is satisfiable.

- **Lemma 19.** *Let ψ be a formula of the form given in (5).*
- *Given $\mathcal{G} \in \mathbb{G}(\psi)$, if $\mathcal{G} \vdash \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$ then $\llbracket \mathcal{G} \rrbracket_{\mathbf{A}} \subseteq \mathcal{B}(\psi)$.*
- *For every $p \in \mathcal{B}(\psi)$, there is $\mathcal{G} \in \mathbb{G}(\psi)$ such that $\mathcal{G} \vdash \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$ and $p \in \llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$.*

Following the case of Büchi arithmetic, we then express $\mathcal{B}(\psi)$ as an ultimately periodic set.

- **Lemma 20.** *Let ψ be as in (5). The set $\mathcal{B}(\psi)$ is ultimately periodic, with threshold bounded by $U^{\mathcal{O}(k \log k)}$ and period bounded by $U^{\mathcal{O}(\ell \cdot k \log k)}$, with $k = n + 3d$ and $\ell = (r + 2) \cdot U^{4n+1}$.*

Together, Proposition 9 and Lemma 20 yield Proposition 12, as we recall that Φ is equivalent to a disjunction $\bigvee_{k \in K} \psi_k$ of formulae ψ_k of the form in (5), with $\#K \leq 2^{\mathcal{O}(\langle \Phi \rangle \log \langle \Phi \rangle)}$.

4 Deciding satisfiability when the base p is large

In view of the magnitude of the bases p established in Theorem 11, in order to prove Theorem 1, i.e., to show that the p -existence and p -universality problems are decidable in NEXP and coNEXP, respectively, it is sufficient to show the following statement.

- **Theorem 21.** *Let Φ be an existential formula of linear arithmetic constraints over p -adic integers (resp. Büchi arithmetic) with parametric base p . Then satisfiability of Φ with respect to a given value $p \in \mathbb{P}$ (resp. $p \geq 2$) can be decided in time $2^{\mathcal{O}(\langle \Phi \rangle^3)} \cdot \mathcal{O}(\langle p \rangle)$.*

This result cannot directly be obtained from [11], where it is shown that satisfiability of Φ with the base p given in binary is decidable NP, as it only gives a coNEXP^{NP} upper bound for p -universality when $\langle p \rangle$ is of exponential size. For our purposes, we require a decision procedure that runs in time polynomial in the size of the binary encoding of p provided as input. This can be achieved by appealing to a strongly polynomial-time algorithm for the feasibility problem of a system of linear Diophantine inequalities in a fixed dimension established in [9].

- **Proposition 22 ([9]).** *Let $S: \mathbf{A} \cdot \mathbf{x} \geq \mathbf{c}$ be a system of linear Diophantine inequalities, with $\mathbf{A} \in \mathbb{Z}^{n \times d}$ and $\mathbf{c} \in \mathbb{Z}^n$. Checking whether $\llbracket S \rrbracket \neq \emptyset$ can be decided using $d^{2.5d+o(d)} \cdot \langle S \rangle$ arithmetic operations, and space polynomial in $\langle S \rangle$.*

With this proposition at hand, proving Theorem 21 for both existential formulas of linear arithmetic constraints over p -adic integers and Büchi arithmetic, respectively, is not difficult. Any such formula can be converted in time $2^{\mathcal{O}(\langle \Phi \rangle \log \langle \Phi \rangle)}$ into a disjunctive normal form with disjuncts of the form given in (2) and (5), respectively. For every disjunct, we iterate in time $2^{\mathcal{O}(\langle \Phi \rangle^2)}$ over all decompositions of the form describe in (4) and (7), respectively, with each decomposition giving rise to family of systems of linear Diophantine equations whose number of variables is bounded by $\mathcal{O}(\langle \Phi \rangle^2)$ and whose coefficients are bounded by $\mathcal{O}(p + 2^{\langle \Phi \rangle})$. Proposition 22 then enables to decide any of such system with the required time bounds. Full details are deferred to Appendix C.

5 Büchi arithmetic: coNEXP lower bound for p -universality

Here, we prove Theorem 2 and show that the p -universality problem for existential Büchi arithmetic is coNEXP-hard. The proof is by a reduction from a coNEXP-complete generalisation of the quantified Boolean satisfiability problem, denoted by $\text{QO}\Pi_1\text{-SAT}$ (where QO stands for “quantified oracle”), that was introduced in [1] and later generalised in [21]. For $m \in \mathbb{N}$, let \mathcal{F}_m denote the set of all m -ary Boolean functions.

The $\text{QO}\Pi_1\text{-SAT}$ problem takes as input a tuple (m, n, φ) where $m, n \in \mathbb{N}$ are written in unary, and φ is a Boolean combination of $x_1, \dots, x_m, y_1, \dots, y_n$ and $f(x_1, \dots, x_m)$. The input is accepted if and only if for all $f \in \mathcal{F}_m$ there are $x_1, \dots, x_m, y_1, \dots, y_n \in \{0, 1\}$ such that φ is a valid.

Our reduction from $\text{QO}\Pi_1\text{-SAT}$ to the p -universality problem of existential Büchi arithmetic follows an approach for showing coNEXP hardness of the Π_2 -fragment of Presburger arithmetic [10, 12]. The main challenge is to show how to universally quantify over and suitably encode the doubly-exponential number of m -ary Boolean functions. Given $f \in \mathcal{F}_m$, we encode f via a number $z \in \mathbb{N}$ using a variant of Gödel encoding as follows:

$$f(b_0, \dots, b_{m-1}) = b \iff z \equiv b \pmod{q}, \text{ for all } q \in \mathbb{P} \cap [k^3, (k+1)^3), k = \sum_{i=0}^{m-1} 2^i \cdot b_i. \quad (9)$$

Note that Ingham’s theorem on prime gaps [15] guarantees that for sufficiently large $k \in \mathbb{N}$, there is at least one prime in the interval $[k^3, (k+1)^3)$. For technical convenience, to avoid adding a constant offset throughout our constructions, and as done in e.g. [10, 12], we apply Ingham’s theorem as if it was true for all $k \in \mathbb{N}$.

► **Lemma 23.** *For every $f \in \mathcal{F}_m$ there is some $z \in \mathbb{N}$ encoding f as specified in (9), and for all $i, j > 0$, p^i is a valid encoding if and only if p^j is a valid encoding.*

Proof. The first part immediately follows from the Chinese remainder theorem, and the second part follows from $b \in \{0, 1\}$ in (9). ◀

From [12] we can derive the existence of the following families of existential Presburger formulas polynomial-time computable in $m \in \mathbb{N}$ given in unary:

- $\Phi_m^{\text{prime}}(x)$ that evaluates to true if and only if $x < 2^m$ and $x \in \mathbb{P}$;
- $\Phi_m^{\text{pow}3}(x, y)$ that evaluates to true if and only if $x < 2^m$ and $y = x^3$;
- $\Phi_m^{\text{mod}}(x, y)$ that evaluates to true if and only if $y < 2^m$ and $x \equiv 0 \pmod{y}$; and
- $\Phi_m^{\text{invalid}}(x)$ that evaluates to true if and only if there are $q_1, q_2 \in \mathbb{P} \cap [k^3, (k+1)^3)$ for some $k < 2^m$ such that $(x \bmod q_1) \neq (x \bmod q_2)$ or $(x \bmod q_1) \notin \{0, 1\}$.

We define a family of existential formulas of Presburger arithmetic $\Phi_m^{\text{fun}}(\mathbf{x}, y, z)$ that hold if and only if $f(x_0, \dots, x_{m-1}) = y$, under the assumptions that z is a valid encoding of some $f \in \mathcal{F}_m$ as defined in Equation (9), $\mathbf{x} = (x_0, \dots, x_{m-1}) \in \{0, 1\}^m$ and $y \in \{0, 1\}$:

$$\begin{aligned} \Phi_m^{\text{fun}}(\mathbf{x}, y, z) \stackrel{\text{def}}{=} \exists x, k, k_0, k_1: & k = \sum_{i=0}^{m-1} 2^i \cdot x_i \wedge \Phi_m^{\text{pow}3}(k, k_0) \wedge \Phi_{m+1}^{\text{pow}3}(k+1, k_1) \\ & \wedge k_0 \leq x < k_1 \wedge \Phi_{3m+1}^{\text{prime}}(x) \wedge \Phi_{3m+1}^{\text{mod}}(z-y, x). \end{aligned}$$

For the final step of our reduction, given an instance $I = (m, n, \varphi)$ of $\text{QO}\Pi_1\text{-SAT}$, denote by $\tilde{\varphi}(\mathbf{x}, \mathbf{y}, y)$ the quantifier-free formula of Presburger arithmetic obtained from φ by replacing x_i and y_i by $x_i = 1$ and $y_i = 1$, respectively; $\neg x_i$ and $\neg y_i$ by $x_i = 0$ and $y_i = 0$, respectively; and $f(x_1, \dots, x_n)$ by $y = 1$ and $\neg f(x_1, \dots, x_n)$ by $y = 0$. We claim that I is a positive instance if and only if the following formula Ψ_p of existential Büchi arithmetic, with parametric base p and a single occurrence of a V_p function, is p -universal:

$$\Psi_p \stackrel{\text{def}}{=} \exists x \exists y \exists \mathbf{x} \exists \mathbf{y} \exists z: V_p(z) = z \wedge z > 1 \wedge 0 \leq x \leq 1 \wedge 0 \leq y \leq 1 \wedge \bigwedge_{i \in [1, m]} 0 \leq x_i \leq 1 \\ \wedge \bigwedge_{i \in [1, n]} 0 \leq y_i \leq 1 \wedge \Phi_m^{\text{fun}}(\mathbf{x}, \mathbf{y}, z) \wedge (\Phi_m^{\text{invalid}}(z) \vee \tilde{\varphi}(\mathbf{x}, \mathbf{y}, y)) ,$$

where $\mathbf{x} = (x_1, \dots, x_m)$ and $\mathbf{y} = (y_1, \dots, y_n)$. Theorem 2 is now an immediate consequence of the following proposition.

► **Proposition 24.** *Let $I = (m, n, \varphi)$ be an instance of $QO\Pi_1$ -SAT. Then I is a positive instance if and only if Ψ_p is p -universal.*

Proof. (\Rightarrow): By Lemma 23, any $f \in \mathcal{F}_m$ is encoded by some $p \in \mathbb{N}$. Choosing $z = p$ in Ψ_p and instantiating the x_j and y_j by those x_j and y_j making φ true for f , which exist by assumption, it follows that Ψ_p is p -universal.

(\Leftarrow): Suppose that Ψ_p is p -universal. By Lemma 23, for every $f \in \mathcal{F}_m$ there is some valid encoding p of f , and by assumption Ψ_p evaluates to true in base p for some choice $z = p^i$. By Lemma 23, p is a valid encoding of f as well, and hence the same x_j and y_j that make Ψ_p for $z = p^i$ and *a fortiori* $z = p$ true also make φ true for f . Hence I is a positive instance. ◀

6 Conclusion

There remains the open problems to what extent the coNEXP upper bound for p -universality for the p -adic integers stated in Theorem 1 is tight. The coNEXP lower bound for p -universality for existential Büchi arithmetic together with the bounds on the ultimately periodic representation of the set of bases satisfying a given formula obtained in Proposition 12 gives strong evidence that, should it be possible to improve the coNEXP upper bound for the p -adic integers, a different approach not based on p -automata will likely be required. Likewise, we do not know whether the NEXP upper bounds for p -existence can be improved.

The coNEXP lower bound for p -universality for Büchi arithmetic crucially relies on the presence of disjunction and conjunctive as Boolean connectives. It would be interesting to better understand the complexity of the conjunctive fragments of the logics we consider, at present we cannot obtain any better upper bounds. In particular, Lechner et al. have shown that the restricted formulas obtained in Lipshitz' decidability proof are p -universal if and only if they are satisfied for all primes p singly exponentially bounded in the input [18].

Another interesting open problem is to settle the decidability status of p -universality for full Büchi arithmetic. Given that Büchi arithmetic does not have quantifier elimination [13], this problem will also likely require new approaches and techniques.

References

- 1 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Comput. Complex.*, 1:3–40, 1991. doi:10.1007/BF01200056.
- 2 Achim Blumensath and Erich Grädel. Automatic structures. In *Logic in Computer Science, LICS*, pages 51–62. IEEE Computer Society, 2000. doi:10.1109/LICS.2000.855755.
- 3 Véronique Bruyère. Entiers et automates finis. *Mémoire de fin d'études*, 1985.
- 4 Véronique Bruyère, Georges Hansel, Christian Michaux, and Roger Villemaire. Logic and p -recognizable sets of integers. *Bull. Belg. Math. Soc. Simon Stevin*, 1(2):191–238, 1994. doi:10.36045/bbms/1103408547.
- 5 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Math. Logic Quart.*, 6(1-6):66–92, 1960. doi:10.1002/ma1q.19600060105.

- 6 Dmitry Chistikov and Christoph Haase. The taming of the semi-linear set. In *International Colloquium on Automata, Languages, and Programming, ICALP*, volume 55 of *LIPICs*, pages 128:1–128:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ICALP.2016.128.
- 7 Alan Cobham. On the base-dependence of sets of numbers recognizable by finite automata. *Math. Syst. Theory*, 3(2):186–192, June 1969.
- 8 Andreas Dolzmann and Thomas Sturm. P-adic constraint solving. In *International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 151–158. ACM, 1999. doi:10.1145/309831.309894.
- 9 András Frank and Éva Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.
- 10 Erich Grädel. Dominoes and the complexity of subclasses of logical theories. *Ann. Pure Appl. Log.*, 43(1):1–30, 1989.
- 11 Florent Guépin, Christoph Haase, and James Worrell. On the existential theories of Büchi arithmetic and linear p -adic fields. In *Logic in Computer Science, LICS*, pages 1–10. IEEE, 2019. doi:10.1109/LICS.2019.8785681.
- 12 Christoph Haase. Subclasses of Presburger arithmetic and the weak EXP hierarchy. In *Computer Science Logic (CSL) and Logic in Computer Science (LICS), CSL-LICS*, pages 47:1–47:10. ACM, 2014. doi:10.1145/2603088.2603092.
- 13 Christoph Haase and Jakub Rózycki. On the expressiveness of Büchi arithmetic. In *Foundations of Software Science and Computation Structures, FOSSACS*, volume 12650 of *Lect. Notes Comp. Sci.*, pages 310–323. Springer, 2021. doi:10.1007/978-3-030-71995-1_16.
- 14 Bernard R. Hodgson. On direct products of automaton decidable theories. *Theor. Comput. Sci.*, 19(3):331–335, 1982. doi:10.1016/0304-3975(82)90042-1.
- 15 Albert E. Ingham. On the Estimation of $N(\sigma, T)$. *Q. J. Math.*, os-11(1):201–202, January 1940. doi:10.1093/qmath/os-11.1.201.
- 16 Bakhadyr Khoushainov and Anil Nerode. Automatic presentations of structures. In *Logical and Computational Complexity, LCC*, volume 960 of *Lect. Notes Comp. Sci.*, pages 367–392. Springer, 1995. doi:10.1007/3-540-60178-3_93.
- 17 Aless Lasaruk and Thomas Sturm. Effective quantifier elimination for Presburger arithmetic with infinity. In *Computer Algebra in Scientific Computing, CASC*, volume 5743 of *Lect. Notes Comp. Sci.*, pages 195–212. Springer, 2009. doi:10.1007/978-3-642-04103-7_18.
- 18 Antonia Lechner, Joël Ouaknine, and James Worrell. On the complexity of linear arithmetic with divisibility. In *Logic in Computer Science, LICS*, pages 667–676. IEEE, 2015. doi:10.1109/LICS.2015.67.
- 19 Yuri V. Linnik. On the least prime in an arithmetic progression. I. The basic theorem. *Rec. Math. [Mat. Sbornik] N.S.*, 15(57):139–178, 1944.
- 20 L. Lipshitz. The diophantine problem for addition and divisibility. *T. Am. Math. Soc.*, 235:271–283, 1978. doi:10.2307/1998219.
- 21 Markus Lohrey. Model-checking hierarchical structures. *J. Comput. Syst. Sci.*, 78(2):461–490, 2012. doi:10.1016/j.jcss.2011.05.006.
- 22 Jaban Meher and M. Ram Murty. Ramanujan’s proof of Bertrand’s postulate. *Am. Math. Mon.*, 120(7):650–653, 2013.
- 23 Loïc Pottier. Minimal solutions of linear Diophantine systems: Bounds and algorithms. In *Rewriting Techniques and Applications, RTA*, volume 488 of *Lect. Notes Comp. Sci.*, pages 162–173. Springer, 1991. doi:10.1007/3-540-53904-2_94.
- 24 Zdeněk Sawa. Efficient construction of semilinear representations of languages accepted by unary nondeterministic finite automata. *Fundam. Informaticae*, 123(1):97–106, 2013. doi:10.3233/FI-2013-802.
- 25 Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Trans. Comput. Theory*, 8(1):3:1–3:36, 2016. doi:10.1145/2858784.

- 26 Aleksei L Semenov. Presburgerness of predicates regular in two number systems. *Sib. Math. J.*, 18(2):289–300, 1977. doi:10.1007/BF00967164.
- 27 Joachim von zur Gathen and Malte Sieveking. A bound on solutions of linear integer equalities and inequalities. *P. Am. Math. Soc.*, 72(1):155–158, 1978. doi:10.2307/2042554.
- 28 Herbert S. Wilf. A circle-of-lights algorithm for the “money-changing problem”. *Am. Math. Mon.*, 85(7):562–565, 1978. doi:10.1080/00029890.1978.11994639.
- 29 Pierre Wolper and Bernard Boigelot. On the construction of automata from linear arithmetic constraints. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, pages 1–19, 2000. doi:10.1007/3-540-46419-0_1.
- 30 Triantafyllos Xylouris. *Über die Nullstellen der Dirichletschen L -Funktionen und die kleinste Primzahl in einer arithmetischen Progression*. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2011. URL: <http://hdl.handle.net/20.500.11811/5074>.

A Missing proofs from Section 2

► **Proposition 6.** For all $w = \mathbf{u}_0 \mathbf{u}_1 \cdots \in (\Sigma_d^p)^\omega$, $\mathbf{A} \cdot \llbracket w \rrbracket^\omega = \mathbf{c}$ iff there is $\mathbf{r} \in \mathbb{Z}^n$ and a strictly ascending sequence $(\lambda_i)_{i \in \mathbb{N}}$ such that $\mathbf{r} \xrightarrow{\mathbf{u}_{\lambda_0-1} \cdots \mathbf{u}_0} \mathbf{A}, p \mathbf{c}$ and $\mathbf{r} \xrightarrow{\mathbf{u}_{\lambda_{j+1}} \cdots \mathbf{u}_{\lambda_j}} \mathbf{A}, p \mathbf{r}$ for all $j \in \mathbb{N}$.

Proof. Simple reformulation of the fact that $\mathbf{A} \cdot \llbracket w \rrbracket^\omega = \mathbf{c}$ holds if and only if for every $k \in \mathbb{N}$ there is $\mathbf{v} \in \mathbb{Z}^n$ such that $\mathbf{A} \cdot \llbracket \mathbf{u}_{k-1} \mathbf{u}_{k-2} \cdots \mathbf{u}_0 \rrbracket^* + \mathbf{v} \cdot p^k = \mathbf{c}$, where $w = \mathbf{u}_0 \mathbf{u}_1, \dots$. Indeed, as the set of live states of the p automaton for $\mathbf{A} \cdot \mathbf{u} = \mathbf{c}$ is finite (by Proposition 4), in the infinite sequence $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \dots$ where $\mathbf{A} \cdot \llbracket \mathbf{u}_{k-1} \mathbf{u}_{k-2} \cdots \mathbf{u}_0 \rrbracket^* + \mathbf{v}_k \cdot p^k = \mathbf{c}$ for every $k \in \mathbb{N}$, there must be a state \mathbf{r} that appears infinitely often. ◀

B Missing proofs from Section 3

Below, given a semi-linear set $M = \bigcup_{i \in I} L(B_i, P_i)$, we define $\|M\| \stackrel{\text{def}}{=} \max_{i \in I} (\|B_i\|_\infty, \|P_i\|_\infty)$.

► **Lemma 14.** Let $\mathbf{A} \in \mathbb{Z}^{n \times d}$, $\mathbf{s}, \mathbf{t} \in \mathbb{Z}^n$ and let T be a linear system of m inequalities. The set $\llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}}$ is an ultimately periodic set with period and threshold bounded by $U^{\mathcal{O}(k \log k)}$, where $k = n + d + m$ and $U = \max(2, \|\mathbf{A}\|_\infty, \|\mathbf{s} \rightarrow_T \mathbf{t}\|_\infty)$.

Proof. Recall that $\llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}}$ is the set of $z \in \mathbb{N}$ for which there is $\mathbf{x} = (x_1, \dots, x_d) \in \llbracket T \rrbracket_{\geq 0}$ satisfying the following system S of linear inequalities:

$$\begin{cases} \mathbf{s} \cdot z + \mathbf{A} \cdot \mathbf{x} = \mathbf{t} \\ z \geq 2 \\ x_i \leq z - 1 \quad \text{for every } i \in [1, d]. \end{cases}$$

As usual, when restricting a system to its non-negative solutions, we can replace inequalities with equalities by introducing slack variables, obtaining the system S' :

$$\begin{cases} \mathbf{s} \cdot z + \mathbf{A} \cdot \mathbf{x} = \mathbf{t} \\ z - z' = 2 \\ x_i + x'_i = z - 1 \quad \text{for every } i \in [1, d]. \end{cases}$$

where z', x'_1, \dots, x'_d are $(d+1)$ variables that shall be interpreted with non-negative integers. It is easy to verify that the set of solutions in $\llbracket S \rrbracket_{\geq 0}$ can be characterised by considering the set $\llbracket S' \rrbracket_{\geq 0}$ and projecting away the dimensions relatives to the slack variables z', x'_1, \dots, x'_d .

Notice that S' has $2 \cdot (d+1)$ variables and $n+d+1$ rows. A similar treatment can be applied to the system of inequalities T : by introducing at most m new slack variables, we obtain a system of equalities T' with m rows and $d+m$ variables. We apply Proposition 7, on the system made of S' and T' and conclude that $\llbracket S' \rrbracket_{\geq 0} \cap \llbracket T' \rrbracket_{\geq 0} = L(B, P)$ where

- $\|B\|_{\infty} \leq ((2 \cdot (d+1) + (d+m) + 2) \cdot U + 1)^{(n+d+1)+(d+m)}$,
- $\|P\|_{\infty} \leq ((2 \cdot (d+1) + (d+m)) \cdot U + 1)^{(n+d+1)+(d+m)}$.

Hence, $\|L(B, P)\| \leq U^{\mathcal{O}(k \log k)}$. Projecting $L(B, P)$ on the variable z yields $\llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}}$. So, $\llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}} = L(C, Q) \subseteq \mathbb{N}$ with $\|L(C, Q)\| \leq U^{\mathcal{O}(k \log k)}$. We now change representation: by Proposition 8, we characterise $\llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}}$ as an ultimately periodic set with period $p \leq \gcd Q \leq U^{\mathcal{O}(k \log k)}$ and threshold $t \leq \|C\|_{\infty} + \|Q\|_{\infty}^2 \leq U^{\mathcal{O}(k \log k)}$. ◀

▶ **Lemma 18.** *The set $\oplus(L_0, \dots, L_r)$ contains all length values generated by \mathcal{G} . One can construct in time $\mathcal{O}(r^2) \cdot U^{\mathcal{O}(n \cdot r)}$ a representation of $\oplus(L_0, \dots, L_r)$ as a semi-linear set $\bigcup_{k \in K} L(\mathbf{b}_k, P_k)$, where $\#K \leq U^{\mathcal{O}(n \cdot r)}$, $\#P \leq r+1$, $\|P\|_{\infty} \leq U^{2n}$ and $\|\mathbf{b}_k\|_{\infty} \leq \mathcal{O}(r \cdot U^{4n})$.*

Proof. It is relatively straightforward to see that $\oplus(L_0, \dots, L_r)$ corresponds to the set of length values generated by \mathcal{G} . First, let $(v_0, \dots, v_r) \in \mathbb{N}^{r+1}$ be a length value generated by \mathcal{G} , which by definition means that \mathcal{G} has a path of the following form

$$\mathbf{t}_{r+1} \rightarrow_{U_r} \mathbf{s}_r \xrightarrow{W_r^{v_r - (v_{r-1} + 1)}} \mathbf{t}_r \rightarrow_{U_{r-1}} \mathbf{s}_{r-1} \dots \mathbf{s}_1 \xrightarrow{W_1^{v_1 - (v_0 + 1)}} \mathbf{t}_1 \rightarrow_{U_0} \mathbf{s}_0 \xrightarrow{W_0^{v_0}} \mathbf{t}_0 = \mathbf{c}.$$

Directly by definition of $G_j(\mathbf{s}_j, \mathbf{t}_j)$ and its related unary NFA, we have that $v_0 \in L_0$ and for every $j \in [1, r]$, $v_j - (v_{j-1} + 1) \in L_j$. By definition of $\oplus(L_1, \dots, L_r)$,

$$\begin{pmatrix} v_0 \\ 1 + v_0 + v_1 - (v_0 + 1) \\ \dots \\ i + v_0 + \sum_{j \in [1, i]} (v_j - (v_{j-1} + 1)) \\ \dots \\ r + v_0 + \sum_{j \in [1, r]} (v_j - (v_{j-1} + 1)) \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ \dots \\ v_i \\ \dots \\ v_r \end{pmatrix} \in \oplus(L_1, \dots, L_r)$$

Conversely, suppose $(v_0, \dots, v_r) \in \oplus(L_1, \dots, L_r)$. This implies that there are ℓ_0, \dots, ℓ_r such that, for every $j \in [0, r]$, $\ell_j \in L_j$ and $v_j = j + \sum_{i \in [0, j]} \ell_i$. By definition of L_j , we conclude that \mathcal{G} has a path of the following form

$$\mathbf{t}_{r+1} \rightarrow_{U_r} \mathbf{s}_r \xrightarrow{W_r^{\ell_r}} \mathbf{t}_r \rightarrow_{U_{r-1}} \mathbf{s}_{r-1} \dots \mathbf{s}_1 \xrightarrow{W_1^{\ell_1}} \mathbf{t}_1 \rightarrow_{U_0} \mathbf{s}_0 \xrightarrow{W_0^{\ell_0}} \mathbf{t}_0 = \mathbf{c}.$$

We have $\ell_0 = v_0$ and, given $j \in [1, r]$, from $v_j = j + \sum_{i \in [0, j]} \ell_i = 1 + v_{j-1} + \ell_j$ we conclude that $\ell_j = v_j - (v_{j-1} + 1)$. Hence (v_0, \dots, v_r) is a length value generated by \mathcal{G} .

Let us now show how to construct a semi-linear representation of $\oplus(L_0, \dots, L_r)$. Recall that, by Proposition 17, for every $i \in [0, r]$ we have $L_i = \bigcup_{j \in J_i} L(b_j, p_j)$ with $\#J_i \leq \mathcal{O}(U^{4n})$, $b_j \leq (2 \cdot U^{2n} + 1) \cdot U^{2n}$ and $p_j \leq U^{2n}$. To compute $\oplus(L_0, \dots, L_r)$ we iterate over all $(j_0, \dots, j_r) \in J_0 \times \dots \times J_r$ and construct a linear set representation for

$$\oplus(L(b_{j_0}, p_{j_0}), \dots, L(b_{j_r}, p_{j_r})) \stackrel{\text{def}}{=} \left\{ \begin{pmatrix} \ell_0 \\ 1 + \ell_0 + \ell_1 \\ \dots \\ r + \sum_{i=0}^r \ell_i \end{pmatrix} \in \mathbb{N}^{r+1} : \forall i \in [0, r], \ell_i \in L(b_{j_i}, p_{j_i}) \right\}.$$

The set $\oplus(L_0, \dots, L_r)$ is then the union over all such linear sets. Hence, consider $(j_0, \dots, j_r) \in J_0 \times \dots \times J_r$ and the linear sets $L(b_{j_0}, p_{j_0}), \dots, L(b_{j_r}, p_{j_r})$. Given $i \in [0, r]$, we write $\mathbf{q}_i \in \mathbb{N}^{r+1}$ for the vector having zero in the first $i-1$ components, and p_{j_i} in the last $r+1-i$ components.

Moreover, let $\mathbf{b} \in \mathbb{N}^{r+1}$ be the vector with i th component set to $i + \sum_{k=0}^i b_{j_k}$. It is easy to see that $\oplus(L(b_{j_0}, p_{j_0}), \dots, L(b_{j_r}, p_{j_r})) = L(\mathbf{b}, \{\mathbf{q}_0, \dots, \mathbf{q}_r\})$. Due to the bound on each $L(b_{j_i}, p_{j_i})$, one can compute $L(\mathbf{b}, \{\mathbf{q}_0, \dots, \mathbf{q}_r\})$ in time $\mathcal{O}(r^2 \cdot \max_{i \in [0, r]} (\langle b_{j_i} \rangle, \langle p_{j_i} \rangle)) \leq \mathcal{O}(n \cdot r^2 \cdot \log U)$. Moreover, $\|\mathbf{q}_i\|_\infty \leq U^{2n}$ and $\|\mathbf{b}\|_\infty \leq \mathcal{O}(r \cdot U^{4n})$. When considering all $U^{\mathcal{O}(n \cdot r)}$ tuples, a semi-linear representation of $\oplus(L_0, \dots, L_r)$ can then be constructed in time $\mathcal{O}(r^2) \cdot U^{\mathcal{O}(n \cdot r)}$. \blacktriangleleft

► **Lemma 19.** *Let ψ be a formula of the form given in (5).*

- *Given $\mathcal{G} \in \mathbb{G}(\psi)$, if $\mathcal{G} \vdash \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$ then $\llbracket \mathcal{G} \rrbracket_{\mathbf{A}} \subseteq \mathcal{B}(\psi)$.*
- *For every $p \in \mathcal{B}(\psi)$, there is $\mathcal{G} \in \mathbb{G}(\psi)$ such that $\mathcal{G} \vdash \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$ and $p \in \llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$.*

Proof. To show this result, we simply adapt the proof of Lemma 15. Let \mathbb{G} be the set of support graphs \mathcal{G} in $\mathbb{G}(\psi)$ such that $\mathcal{G} \vdash \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$. The lemma equivalently states that $\mathcal{B}(\psi) = \bigcup_{\mathcal{G} \in \mathbb{G}} \llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$. Below, we refer to the first and second points in the lemma as the two inclusions \supseteq and \subseteq of this equality.

(\supseteq): Let \mathcal{G} be a support graph in \mathbb{G} , and consider $p \in \llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$. Moreover, let $(v_0, \dots, v_r) \in \mathbb{N}^{r+1}$ be a length value generated by \mathcal{G} such that $\mathbf{B} \cdot \mathbf{c} \geq \mathbf{d} \wedge \bigwedge_{i \in [0, r]} x_i = v_i$ is satisfiable. This means that \mathcal{G} has a path of the form

$$\mathbf{t}_{r+1} \rightarrow_{U_r} \mathbf{s}_r \xrightarrow{v_r - (v_{r-1} + 1)}_{W_r} \mathbf{t}_r \rightarrow_{U_{r-1}} \mathbf{s}_{r-1} \dots \mathbf{s}_1 \xrightarrow{v_1 - (v_0 + 1)}_{W_1} \mathbf{t}_1 \rightarrow_{U_0} \mathbf{s}_0 \xrightarrow{v_0}_{W_0} \mathbf{t}_0 = \mathbf{c}$$

Moreover, as $\mathcal{G} \in \mathbb{G}(\psi)$, there is $\mathbf{r} \in \mathbb{Z}^n$ such that \mathcal{G} has a path from \mathbf{r} to \mathbf{t}_{r+1} as well as a path from \mathbf{r} to itself. We recall that, by definition, for every edge $\mathbf{s} \rightarrow_T \mathbf{t}$ of \mathcal{G} there is $\mathbf{u} \in \Sigma_p^d$ such that $\mathbf{s} \xrightarrow{\mathbf{u}}_{\mathbf{A}, p} \mathbf{t}$ and $\mathbf{u} \in \llbracket T \rrbracket$. We conclude that there is an infinite word w such that $w = w_0 \cdot \mathbf{v}_0 \cdot w_1 \cdot \dots \cdot w_r \cdot \mathbf{v}_r \cdot \bar{w}$, where

1. $\mathbf{v}_0, \dots, \mathbf{v}_r \in \Sigma_p^d$, and for all $j \in [0, r]$, the prefix $w_0 \cdot \mathbf{v}_0 \cdot \dots \cdot w_j \in (\Sigma_p^d)^*$ of w has length v_j ,
2. $\mathbf{t}_{r+1} \xrightarrow{v_r}_{\mathbf{A}, p} \mathbf{s}_r \dots \mathbf{s}_1 \xrightarrow{(w_1)^R}_{\mathbf{A}, p} \mathbf{t}_1 \xrightarrow{v_0}_{\mathbf{A}, p} \mathbf{s}_0 \xrightarrow{(w_0)^R}_{\mathbf{A}, p} \mathbf{t}_0 = \mathbf{c}$,
3. $\bar{w} = \mathbf{u}_0 \mathbf{u}_1 \dots$ is an infinite suffix of w for which there is an infinite sequence $\lambda_0 < \lambda_1 < \dots$ such that $\mathbf{r} \xrightarrow{\mathbf{u}_{\lambda_0 - 1} \dots \mathbf{u}_0}_{\mathbf{A}, p} \mathbf{c}$ and for all $j \in \mathbb{N}$, $\mathbf{r} \xrightarrow{\mathbf{u}_{\lambda_{j+1}} \dots \mathbf{u}_{\lambda_j}}_{\mathbf{A}, p} \mathbf{r}$,
4. By definition of U_0, \dots, U_m and W_0, \dots, W_{m+1} , given $(i, j) \in J$, so that $v_p(u_i) = x_j$ appears in ψ , we have that $w_0 \cdot \mathbf{v}_0 \cdot \dots \cdot w_j$ projected on the encoding of u_i is in $\{0\}^*$, and v_j projected on the encoding of u_i is in $[1, p - 1]$.

Therefore, ψ admits a solution $(\llbracket w \rrbracket^\omega, \mathbf{x})$, where in \mathbf{x} for every $i \in [0, r]$ we have $x_i = v_i$. Indeed, from points 2 and 3, and by Proposition 6, $\mathbf{A} \cdot \llbracket w \rrbracket^\omega = \mathbf{c}$. From points 1 and 4, given $(i, j) \in J$, the l sd encoding of u_i is such that $v_p(u_i) = v_j$, hence $v_p(u_i) = x_j$. Lastly, by definition of (v_0, \dots, v_r) we have $x_0 < x_1 < \dots < x_r$, and we know that $\mathbf{B} \cdot \mathbf{c} \geq \mathbf{d} \wedge \bigwedge_{i \in [0, r]} x_i = v_i$ is satisfiable. This means that ψ is satisfiable with respect to the base p , i.e. $p \in \mathcal{B}(\psi)$.

(\subseteq): Follows conversely from the other inclusion. Briefly, suppose ψ satisfiable with respect to the base p . Consider a word $w \in (\Sigma_p^d)^\omega$ and $\mathbf{x} \in \mathbb{Z}^e$ such that $(\llbracket w \rrbracket^\omega, \mathbf{x})$ is a solution of ψ . As already said in the body of the paper, this means that w must have a prefix of the form $w_0 \cdot \mathbf{v}_0 \cdot w_1 \cdot \dots \cdot w_r \cdot \mathbf{v}_r \cdot w_{r+1}$ such that $\mathbf{v}_0, \dots, \mathbf{v}_r \in \Sigma_p^d$, the word $w_0 \in (\Sigma_p^d)^*$ has length x_0 , each $w_i \in (\Sigma_p^d)^*$ with $i \in [1, r]$ has length $x_i - (x_{i-1} + 1) \geq 0$,

$$\mathbf{r} = \mathbf{s}_{r+1} \xrightarrow{(w_{r+1})^R}_{\mathbf{A}, p} \mathbf{t}_{r+1} \xrightarrow{v_r}_{\mathbf{A}, p} \mathbf{s}_r \dots \mathbf{s}_1 \xrightarrow{(w_1)^R}_{\mathbf{A}, p} \mathbf{t}_1 \xrightarrow{v_0}_{\mathbf{A}, p} \mathbf{s}_0 \xrightarrow{(w_0)^R}_{\mathbf{A}, p} \mathbf{t}_0 = \mathbf{c}$$

and \mathbf{r} is a live state of $\mathcal{A}_p^*(S)$ for which the ω -regular condition of Proposition 6 is satisfied. Moreover, for every $j \in [0, r]$ the values of v_j for the variables $u_1, \dots, u_{\#I}$ satisfy the system U_j , whereas at each position of the word w_j ($j \in [0, r + 1]$) shall satisfy the system W_j .

■ **Algorithm 1** Procedure for deciding satisfiability of a formula ψ of the form in (5).

```

1: function SAT( $\psi$ )
2:    $Q \leftarrow [-U, U]^n$  where  $U = \max(\|\mathbf{A}\|_{1,\infty}, \|\mathbf{c}\|_\infty)$ 
3:   for  $(\mathbf{s}_0, \mathbf{t}_0, \dots, \mathbf{s}_{r+1}, \mathbf{t}_{r+1}) \in Q^{2(r+1)}$  with  $\mathbf{t}_0 = \mathbf{c}$  do
4:      $L_0, \dots, L_r \leftarrow \emptyset$ 
5:     for  $j \in [0, r+1]$  do ▷ below,  $W_j$  and  $U_j$  defined as in Section 3
6:        $\mathcal{A}_j \leftarrow (Q, \delta, \{\mathbf{s}_j\}, \{\mathbf{t}_j\})$  where  $\delta = \{(\mathbf{s}, \mathbf{t}) : p \in \llbracket \mathbf{s} \rightarrow_{W_j} \mathbf{t} \rrbracket_{\mathbf{A}}, \mathbf{s}, \mathbf{t} \in Q\}$ .
7:       if  $j > 0$  then check  $p \in \llbracket \mathbf{t}_j \rightarrow_{U_j} \mathbf{s}_{j-1} \rrbracket_{\mathbf{A}}$ 
8:       check if  $\mathcal{A}_j$  has a path form  $\mathbf{s}_j$  to  $\mathbf{t}_j$  (i.e.  $\mathcal{L}(\mathcal{A}_j) \neq \emptyset$ )
9:       if  $j = r+1$  then check if  $\mathcal{A}_j$  has a non-empty path form  $\mathbf{s}_j$  to itself
10:      if  $j \neq r+1$  then  $L_j \leftarrow L$  where  $L$  defined as in Proposition 17 w.r.t.  $\mathcal{A}_j$ 
11:    end for
12:    if  $\oplus(L_0, \dots, L_r) \cap \llbracket \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d} \rrbracket \neq \emptyset$  then return true ▷  $\oplus(L_0, \dots, L_r)$  as in (8)
13:  end for
14:  return false

```

Let \mathcal{G} be the support graph with edges $\mathbf{t}_1 \rightarrow_{U_0} \mathbf{s}_0, \dots, \mathbf{t}_{r+1} \rightarrow_{U_j} \mathbf{s}_r$ together with $\mathbf{i} \rightarrow_{W_j} \mathbf{i}'$, for every $j \in [0, r+1]$ and every two states \mathbf{i}, \mathbf{i}' such that $\mathbf{i} \rightarrow_{\mathbf{A}, p} \mathbf{i}'$ appears in the path going from \mathbf{s}_j to \mathbf{t}_j , and $\mathbf{i} \rightarrow_{W_{r+1}} \mathbf{i}'$, for every two states \mathbf{i}, \mathbf{i}' such that $\mathbf{i} \rightarrow_{\mathbf{A}, p} \mathbf{i}'$ appears in the path going from \mathbf{r} to itself. By definition, $\mathcal{G} \in \mathbb{G}(\psi)$ and $p \in \llbracket \mathcal{G} \rrbracket_{\mathbf{A}}$. Moreover, (x_0, \dots, x_r) is a length value generated by \mathcal{G} , which entails $\mathcal{G} \vdash \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$. ◀

► **Lemma 20.** *Let ψ be as in (5). The set $\mathcal{B}(\psi)$ is ultimately periodic, with threshold bounded by $U^{\mathcal{O}(k \log k)}$ and period bounded by $U^{\mathcal{O}(\ell \cdot k \log k)}$, with $k = n + 3d$ and $\ell = (r+2) \cdot U^{4n+1}$.*

Proof. Analogous to the proof of Lemma 16. Consider the family of support graphs $\mathbb{G}(\psi)$. Since $\mathbb{G}(\psi)$ is finite, according to Lemma 19 we have $\mathcal{B}(\psi) = \bigcup_{G \in \mathbb{G}} \llbracket G \rrbracket_{\mathbf{A}}$ for some $G \subseteq \mathbb{G}(\psi)$. Every edge $\mathbf{s} \rightarrow_T \mathbf{t}$ of a support graph $\mathcal{G} \in G$ is such that $\|\mathbf{s}\|_\infty, \|\mathbf{t}\|_\infty \leq \max(\|\mathbf{A}\|_{1,\infty}, \|\mathbf{c}\|_\infty)$ and T is a system among $U_0, \dots, U_r, W_0, \dots, W_{r+1}$. Hence, all the graphs in G are built from a set E of $(2 \cdot \max(\|\mathbf{A}\|_{1,\infty}, \|\mathbf{c}\|_\infty))^{2n} \cdot (2r+3) \leq \ell$ edges. Each possible linear system T labelling an edge in E has at most $2d$ inequalities, with coefficients and constants in $\{0, 1\}$. By Lemma 14, each edge $e \in E$ is such that $\llbracket e \rrbracket_{\mathbf{A}}$ is an ultimately periodic set with period and threshold bounded by $U^{\mathcal{O}(k \log k)}$, where $k = n + 3d$. By Proposition 9, $\mathcal{B}(\psi) = \bigcup_{G \in \mathbb{G}} \llbracket G \rrbracket_{\mathbf{A}}$ is an ultimately periodic set with threshold in $U^{\mathcal{O}(k \log k)}$ and period in $U^{\mathcal{O}(\ell \cdot k \log k)}$. ◀

C Missing proofs from Section 4

► **Theorem 21.** *Let Φ be an existential formula of linear arithmetic constraints over p -adic integers (resp. Büchi arithmetic) with parametric base p . Then satisfiability of Φ with respect to a given value $p \in \mathbb{P}$ (resp. $p \geq 2$) can be decided in time $2^{\mathcal{O}(\langle \Phi \rangle^3)} \cdot \mathcal{O}(\langle p \rangle)$.*

Proof. Consider an existential formula Φ of linear arithmetic constraints over p -adic integers, with parametric base q , and a concrete value $p \in \mathbb{P}$ for q . As done in Proposition 12, in time exponential in $\langle \Phi \rangle$ we manipulate Φ and obtain a disjunctive normal form where each of the $2^{\mathcal{O}(\langle \Phi \rangle \log \langle \Phi \rangle)}$ disjuncts are of size $\mathcal{O}(\langle \Phi \rangle)$ and the form in (5). We then iterate through each disjunct ψ of Φ , calling the procedure SAT(ψ) described in Algorithm 1. If the procedure returns true for some disjunct ψ , then Φ is satisfiable, otherwise it is unsatisfiable. Let us fix a disjunct ψ of Φ . We argue that SAT(ψ) is a decision procedure for the satisfiability problem of ψ that runs in time $2^{\mathcal{O}(\langle \psi \rangle^3)} \cdot \mathcal{O}(\langle p \rangle)$. As Φ has $2^{\mathcal{O}(\langle \Phi \rangle \log \langle \Phi \rangle)}$ many disjuncts of size $\mathcal{O}(\langle \Phi \rangle)$, this is sufficient to establish Theorem 21.

Let us sketch the correctness of the algorithm, which essentially follows by replaying some of the arguments that led to Proposition 12. Indeed, line 3 iterates through all possible tuples of intermediate live states, as done when considering the set of automatic support graphs $\mathbb{G}(\psi)$. Let $(\mathbf{s}_0, \mathbf{t}_0, \dots, \mathbf{s}_{r+1}, \mathbf{t}_{r+1})$ be one of these tuples. For every $j \in [0, r+1]$, the algorithm considers the unary NFA \mathcal{A}_j (line 6) whose arrows (\mathbf{s}, \mathbf{t}) corresponds to edges $\mathbf{s} \rightarrow_{W_j} \mathbf{t}$ of automatic support graphs for which $p \in \llbracket \mathbf{s} \rightarrow_{W_j} \mathbf{t} \rrbracket_{\mathbf{A}}$. The fundamental relation between \mathcal{A}_j and the p -automaton $\mathcal{A}_p^*(S)$ for the system $S: \mathbf{A} \cdot \mathbf{u} = \mathbf{c}$ is as follows:

$$(\mathbf{s}, \mathbf{t}) \in \delta \text{ if and only if there is } \mathbf{u} \in [0, p-1]^n \text{ such that } \mathbf{s} \xrightarrow{\mathbf{u}}_{\mathbf{A}, p} \mathbf{t} \text{ and } \mathbf{u} \in \llbracket W_j \rrbracket.$$

Therefore, if the checks performed in lines 7 and 8 are satisfied for all $j \in [0, r+1]$ (else, a new tuple in $Q^{2(r+1)}$ is considered), we conclude that $\mathcal{A}_p^*(S)$ contains a path of the form in (6). According to Proposition 6, line 9 then tests for the existence of a non-empty path in \mathcal{A}_{r+1} (equivalently, $\mathcal{A}_p^*(S)$) going from \mathbf{s}_{r+1} to itself. If such a path is found, then there is an infinite word $w \in (\Sigma_p^d)^\omega$ and $\mathbf{x} \in \mathbb{N}^e$ such that $(\llbracket w \rrbracket^*, \mathbf{x})$ is a solution for the subformula $\mathbf{A} \cdot \mathbf{u} = \mathbf{c} \wedge \bigwedge_{(i,j) \in J} v_p(u_i) = x_j \wedge \bigwedge_{j \in [1,r]} x_{j-1} < x_j$ of ψ . Lines 10 and 12 provides further analysis needed to check for the satisfaction of the system $\mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$. Again following what done for Proposition 12, line 10 computes the set L_j of lengths of words accepted by \mathcal{A}_j , that are then combined in the set $\oplus(L_0, \dots, L_r)$. Note that $\oplus(L_0, \dots, L_r)$ is empty if so is one set among L_0, \dots, L_r . If one element of $\oplus(L_0, \dots, L_r)$ satisfies $\mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$, then ψ is satisfiable, and the procedure returns true (line 12).

Let us discuss the time complexity of $\text{SAT}(\psi)$. First of all, the running time of the loop in line 5 is in $2^{\mathcal{O}(\langle \psi \rangle^2)} \cdot \mathcal{O}(\langle p \rangle)$. Indeed, notice that given $\mathbf{s}, \mathbf{t} \in Q$ and a system $T \in \{U_j, W_j, W_{r+1} : j \in [0, r]\}$, the membership problem $p \in \llbracket \mathbf{s} \rightarrow_T \mathbf{t} \rrbracket_{\mathbf{A}}$ correspond to the non-emptiness problem $\llbracket R \rrbracket \neq \emptyset$ of the linear system $R: \mathbf{t} = \mathbf{s} \cdot p + \mathbf{A} \cdot \mathbf{x} \wedge \|\mathbf{x}\| < p \wedge \mathbf{x} \in \llbracket T \rrbracket$, which by Proposition 22 can be decided in time $\langle \psi \rangle^{\mathcal{O}(\langle \psi \rangle)} \cdot \mathcal{O}(\langle p \rangle)$. Hence, by iterating over all pairs $(\mathbf{s}, \mathbf{t}) \in Q^2$, one constructs the automaton \mathcal{A}_j in time $2^{\mathcal{O}(\langle \psi \rangle^2)} \cdot \mathcal{O}(\langle p \rangle)$ (line 6). Line 7 runs in time $\langle \psi \rangle^{\mathcal{O}(\langle \psi \rangle)} \cdot \mathcal{O}(\langle p \rangle)$. Lines 8 and 9, can be implemented with a depth-first search on the automaton \mathcal{A}_j , which takes time bounded by the cardinality of the transition relation δ , that is bounded by $2^{\mathcal{O}(\psi)^2}$. Directly from Proposition 17, line 10 can be performed in time $2^{\mathcal{O}(\langle \psi \rangle^2)}$. Let us now look at line 12. By Lemma 18, computing $\oplus(L_0, \dots, L_r)$ as a semi-linear set can be done in time $2^{\mathcal{O}(\langle \psi \rangle^3)}$. This set is of the form $\bigcup_{k \in K} L(\mathbf{b}_k, P_k) \subseteq \mathbb{N}^{r+1}$, with $\#K \leq 2^{\mathcal{O}(\langle \psi \rangle^3)}$, $\|\mathbf{b}_k\|_\infty, \|P_k\|_\infty \leq 2^{\mathcal{O}(\langle \psi \rangle^2)}$ and $\#P_k \leq r+1$. Hence, to check for the non-emptiness of the intersection in line 12, it is sufficient to iterate over all $k \in K$ and check for the satisfiability of the system $\mathbf{y} = \mathbf{b}_k + P_k \cdot \boldsymbol{\lambda} \wedge \mathbf{B} \cdot \mathbf{x} \geq \mathbf{d}$, where \mathbf{y} is a subset of the variables appearing in \mathbf{x} . By Proposition 22, this can be done in time $\langle \psi \rangle^{\mathcal{O}(\langle \psi \rangle)}$. Overall, line 12 can be evaluated in $2^{\mathcal{O}(\langle \psi \rangle^3)}$, and hence the running time for the body of the for loop of line 3 is bounded by $2^{\mathcal{O}(\langle \psi \rangle^3)} \cdot \mathcal{O}(\langle p \rangle)$. As this loop iterates over $(2U)^{2(r+1)} \leq 2^{\mathcal{O}(\langle \psi \rangle^2)}$ tuples, the running time of $\text{SAT}(\psi)$ is in $2^{\mathcal{O}(\langle \psi \rangle^3)} \cdot \mathcal{O}(\langle p \rangle)$. ◀

In order to establish Theorem 21 for the case of existential Büchi arithmetic, it is sufficient to bring Φ into disjunctive normal form with disjuncts of the form in Equation (2), and call on each disjunct ψ the procedure $\text{SAT}(\psi)$ subject to the following updates: (I) reflecting (3), the iteration on line 3 is on tuples such that $\mathbf{s}_0 = \mathbf{0}$ and $\mathbf{t}_{r+1} = \mathbf{c}$, and the test in line 7 becomes $j > 0$ and $p \notin \llbracket \mathbf{t}_{j-1} \rightarrow_{U_j} \mathbf{s}_j \rrbracket_{\mathbf{A}}$; (II) the systems U_j and W_j in lines 6 and 7 are defined as introduced in Section 3 and (III) lines 4, 9, 10 and 12 are removed. The proof of Theorem 21 can be easily updated accordingly.

Obstructing Classification via Projection

Pantea Haghikhatkhan ✉

TU Eindhoven, The Netherlands

Wouter Meulemans ✉

TU Eindhoven, The Netherlands

Bettina Speckmann ✉ 

TU Eindhoven, The Netherlands

Jérôme Urhausen ✉

Utrecht University, The Netherlands

Kevin Verbeek ✉

TU Eindhoven, The Netherlands

Abstract

Machine learning and data mining techniques are effective tools to classify large amounts of data. But they tend to preserve any inherent bias in the data, for example, with regards to gender or race. Removing such bias from data or the learned representations is quite challenging. In this paper we study a geometric problem which models a possible approach for bias removal. Our input is a set of points P in Euclidean space \mathbb{R}^d and each point is labeled with k binary-valued properties. A priori we assume that it is “easy” to classify the data according to each property. Our goal is to obstruct the classification according to one property by a suitable projection to a lower-dimensional Euclidean space \mathbb{R}^m ($m < d$), while classification according to all other properties remains easy.

What it means for classification to be easy depends on the classification model used. We first consider classification by linear separability as employed by support vector machines. We use Kirchberger’s Theorem to show that, under certain conditions, a simple projection to \mathbb{R}^{d-1} suffices to eliminate the linear separability of one of the properties whilst maintaining the linear separability of the other properties. We also study the problem of maximizing the linear “inseparability” of the chosen property. Second, we consider more complex forms of separability and prove a connection between the number of projections required to obstruct classification and the Helly-type properties of such separabilities.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Models of learning

Keywords and phrases Projection, classification, models of learning

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.56

Funding *Jérôme Urhausen*: Supported by the Dutch Research Council (NWO); 612.001.651.

Acknowledgements Research on the topic of this paper was initiated at the 5th Workshop on Applied Geometric Algorithms (AGA 2020) in Langbroek, NL. The authors thank Jordi Vermeulen for initial discussions on the topic of this paper.

1 Introduction

Classification is one of the most basic data analysis operators: given a (very) large set of high-dimensional input data with a possibly large set of heterogeneous properties, we would like to classify the data according to one or more of these properties to facilitate further analysis and decision making. Machine learning and data mining techniques are frequently employed in this setting, since they are effective tools to classify large datasets. However, just as any data-driven techniques, they tend to preserve any bias inherent in the data,



© Pantea Haghikhatkhan, Wouter Meulemans, Bettina Speckmann, Jérôme Urhausen, and Kevin Verbeek;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 56; pp. 56:1–56:19

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for example, with regards to gender or race. Such bias arises from under-representation of minority groups in the data or is caused by historical data, which reflect outdated societal norms. Bias in the data might be inconsequential, for example in music recommendations, but it can be harmful when classification algorithms are used to make life-changing decisions on, for example, loans, recruitment, or parole [23].

Naturally, the identification and removal of bias receives a significant amount of attention, although the problem is still far from solved. For example, Mehrabi et al. [19] provide a taxonomy of fairness definitions and bias types. They list the biases caused by data and the types of discrimination caused by machine learning techniques. Many approaches have been considered to eliminate or reduce bias in machine learning models. Some researchers have used a statistical approach to address this problem (e.g., [13]), while others focus on data preprocessing or controlling the sampling to compensate for bias or under-representation in the data (e.g., [2, 15]). Another approach is to use an additional (adversarial) machine learning model to eliminate bias in the first model (e.g., [11, 18, 27]). One major problem of attempting to eliminate bias (or increasing fairness) in machine learning is that it may negatively affect the accuracy of the learned model. This trade-off has also been studied extensively (e.g., [3, 25]).

We are particularly interested in data that is represented by vectors in high-dimensional Euclidean space. Such data arises, for example, from word embeddings for textual data. Several studies show that the bias present in the training corpora is also present in the learned representation (e.g. [7, 8]). Abbasi et al. [1] recently introduced a geometric notion of stereotyping. In this paper we follow the same premise that bias is in some form encoded in the geometric or topological features of the high-dimensional vector representation and that manipulating this geometry can remove the bias. This premise has been the basis for many papers on algorithmic fairness (e.g., [11, 12, 26]).

Several papers investigate the theory that gender is captured in certain dimensions of the data. Bolukbasi et al. [5] postulate that the bias manifests itself in specific “particularly gendered” words and that equalizing distances to these special words removes bias. Zhao et al. [28] devise a model which attempts to represent gender in one dimension which can be removed after training to arrive at a (more) gender-neutral word representation. Bordia and Bowman [6] remove bias by minimizing the projection of the embeddings on the gender subspace (using a regularization term in the training process). Very recently, various papers [9, 10, 14, 22] explored the direct use of projection to remove sensitive properties of the data. In some cases the data is not projected completely, as removing sensitive properties completely may negatively affect the quality of the model.

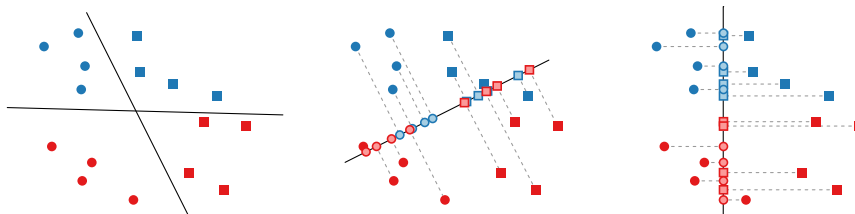
In this paper we take a slightly more general point of view. We say that a property is present in the data representation if it is “easy” to classify the data according to that property. That is, a property (such as gender) can be described by more complicated geometric relations than a subspace. Given the premise that the geometry of word embeddings encodes important relations between the data, then any bias removal technique needs to preserve as much as possible of these relations. Hence we investigate the use of projection to eliminate bias while maintaining as many other relations as possible. We say that the relation of data points with respect to specific properties is maintained by a projection, if it is still easy to classify according to these properties after projection. Our paper explores how well projection can obstruct classification according to a specific property (such as gender) for certain classification models.

Problem statement. Our input is a set of n points $P = \{p_1, \dots, p_n\}$ in general position in \mathbb{R}^d . For convenience we identify the points with their corresponding vector. We model the various properties of the data (such as gender) as binary labels.¹ Hence, for all points in P we are also given k binary-valued *properties*, represented as functions $a_i: P \rightarrow \{-1, 1\}$ for $1 \leq i \leq k$. We denote the subset of points $p \in P$ with $a_i(p) = 1$ as P_+^i , and the subset of points $p \in P$ with $a_i(p) = -1$ as P_-^i for $1 \leq i \leq k$. For a point $p \in P$, we refer to the tuple $(a_1(p), \dots, a_k(p))$ as the *label* of p . Note that there are 2^k different possible labels. Generally speaking, we do not know which specific properties a dataset has. However, to study the influence of projection on all relevant properties of a dataset, we assume that these properties are given.

We assume that it is “easy” to classify the points in P according to the properties by using the point coordinates. Throughout the paper, we consider different definitions for what is considered easy or difficult to classify. Our goal is to compute a projection P' of P to lower dimensions such that the first property a_1 becomes difficult to classify in P' , and the other properties a_2, \dots, a_k remain easy to classify in P' . As a shorthand we use the notation $P_- = P_-^1$ and $P_+ = P_+^1$ for the point sets in which the special property a_1 is set to -1 and $+1$, respectively. Similarly, we use the notation P'_- and P'_+ for the point sets P_- and P_+ after projection. In most cases we will consider a projection along a single unit vector w ($\|w\| = 1$), mapping points in \mathbb{R}^d to points in \mathbb{R}^{d-1} . For a point $p_i \in P$, we denote its projection as $p'_i = p_i - (p_i \cdot w)w$, where $(p_i \cdot w)$ denotes the dot product between the vectors $p_i, w \in \mathbb{R}^d$. To assign coordinates to p'_i in \mathbb{R}^{d-1} , we need to establish a basis for the projected space. We therefore often consider p'_i to lie in the original space \mathbb{R}^d , where the coordinates of p'_i are restricted to the hyperplane that is orthogonal to w and passes through the origin. Sometimes we will consider projections along multiple vectors w_1, \dots, w_r . In that case we assume that $\{w_j\}_{j=1}^r$ form an orthonormal system, such that we can write the projection as $p'_i = p_i - \sum_{j=1}^r (p_i \cdot w_j)w_j$. Again, we assume that p'_i still lies in \mathbb{R}^d , but is restricted to the $(d - r)$ -dimensional flat that is orthogonal to w_1, \dots, w_r and passes through the origin.

We consider different models for defining what is easy or difficult to classify, resulting in different computational problems. These models typically rely on a form of “separability” between two point sets. For a specific definition of separability, using a slight abuse of notation, we will often state that a property a_i is separated in a point set P when we actually mean that P_-^i and P_+^i are separated (see Figure 1 for a simple example in \mathbb{R}^2). The specific models, along with the relevant definitions, are described in detail in the respective sections.

¹ Neither gender nor many other societally relevant properties are binary, however, we restrict ourselves to binary properties to simplify our mathematical model.



■ **Figure 1** Left: data points with two linearly-separable properties: shape and color. Middle: a projection which keeps shape separated, but not color. Right: a projection with the opposite effect.

Contributions and organization. In Section 2 we consider linear separability as the classification model. We first show that, if even one possible label is missing from P , then there may be no projection that eliminates the linear separability of a_1 whilst keeping the linear separability of the other properties. On the other hand, if all possible labels are present in the point set, then we show that it is always possible to achieve this goal. In Appendix A we discuss a related question: given a measure to quantify how far removed a labeled point set is from linear separability, how can we optimize this measure for a_1 after projection? We show that the optimal projection can be computed efficiently under certain specific conditions, but may be hard to compute efficiently in general. In Section 3 we introduce (b, c) -separability, which is a generalization of linear separability. Although a single projection is no longer sufficient to avoid (b, c) -separability of a_1 after projection, we show that, in general, the number of projections needed to achieve this is linked to the Helly number of the respective separability predicate. We then establish bounds on the Helly numbers of (b, c) -separability for specific values of b and c . Omitted proofs can be found in Appendix B.

2 Linear separability

In this section we consider linear separability for classification. For a point set P and property $a_i: P \rightarrow \{-1, 1\}$, we say that a_i is easy to classify on P if P_-^i and P_+^i are (strictly) linearly separable; we say that a_i is difficult to classify otherwise. Two point sets P and Q ($P, Q \subset \mathbb{R}^d$) are *linearly separable* if there exists a hyperplane H separating P from Q . The point sets are *strictly linearly separable* if we can additionally require that none of the points lie on H . Equivalently, the point sets P and Q are linearly separable if there exists a unit vector $v \in \mathbb{R}^d$ and constant $c \in \mathbb{R}$ such that $(v \cdot p) \leq c$ for all $p \in P$ and $(v \cdot q) \geq c$ for all $q \in Q$ (v is the normal vector of the hyperplane H). We say that P and Q are linearly separable *along* v . If the inequalities can be strict, then the point sets are strictly linearly separable.

One of the machine learning techniques that use linear separability for classification are *support vector machines* (SVMs). SVMs compute the (optimal) hyperplane that separates two classes in the training data (if linearly separable), and use that hyperplane for further classifications. Linear separability is therefore a good first model to consider for classification.

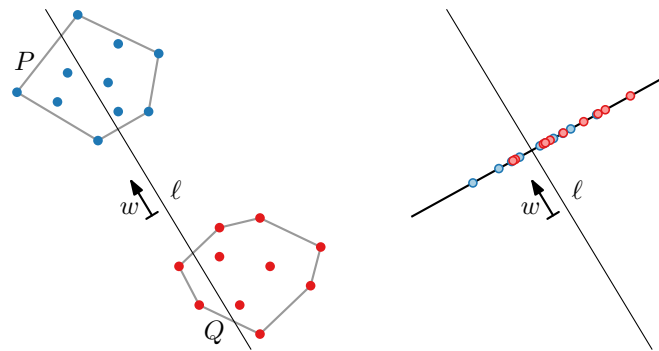
Let $CH(P)$ denote the convex hull of a point set P . By definition, we have that $x \in CH(P)$ if and only if there exist coefficients $\lambda_i \geq 0$ such that $x = \sum_{i=1}^n \lambda_i p_i$ and $\sum_{i=1}^n \lambda_i = 1$. We use the following basic results on convex geometry and linear algebra.

► **Fact 1.** *Two point sets P and Q are linearly separable iff $CH(P)$ and $CH(Q)$ are interior disjoint. P and Q are strictly linearly separable iff $CH(P) \cap CH(Q) = \emptyset$.*

► **Observation 2.** *Let $P' = \{p'_1, \dots, p'_n\}$ be the point set obtained from $P = \{p_1, \dots, p_n\}$ by projecting along a unit vector w . If $x = \sum_{i=1}^n \lambda_i p_i$ (for $\lambda_i \in \mathbb{R}$), then $x' = x - (w \cdot x)w = \sum_{i=1}^n \lambda_i p'_i$. Specifically, if $x \in CH(P)$, then $x' \in CH(P')$.*

► **Lemma 3.** *Let P and Q be two point sets. If we project both P and Q along a unit vector w to obtain P' and Q' , then P' and Q' are not strictly linearly separable iff there exists a line ℓ parallel to w that intersects both $CH(P)$ and $CH(Q)$. If ℓ intersects the interior of $CH(P)$ or $CH(Q)$, then P' and Q' are not linearly separable.*

Proof. Assume that the line ℓ exists, and it contains $x_P \in CH(P)$ and $x_Q \in CH(Q)$ (see Figure 2). By construction, $x' = x_P - (w \cdot x_P)w = x_Q - (w \cdot x_Q)w$. Hence, by Observation 2, $x' \in CH(P') \cap CH(Q')$. Thus, by Fact 1, P' and Q' are not strictly linearly separable.

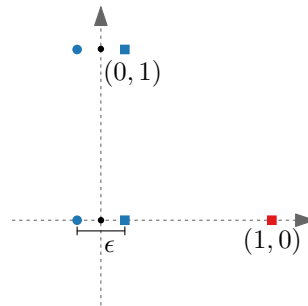


■ **Figure 2** Line ℓ intersects $CH(P)$ and $CH(Q)$; after projection the convex hulls intersect.

For the other direction, choose $x' \in CH(P') \cap CH(Q')$. The line parallel to w and passing through x' must clearly intersect both $CH(P)$ and $CH(Q)$. The extension to (non-strict) linear separability is straightforward. ◀

Assume now that the properties a_1, \dots, a_k are strictly linearly separable in P . Can we project P along a unit vector w so that a_2, \dots, a_k are still strictly linearly separable in P' , but a_1 is not? We consider two variants: (1) *separation preserving* and (2) *separability preserving* projections. The former preserves a fixed set of separating hyperplanes H_2, \dots, H_k for properties a_2, \dots, a_k , the latter preserves only linear separability of a_2, \dots, a_k .

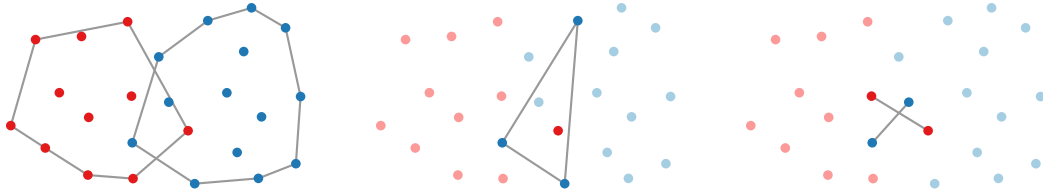
Lemma 4 proves there exist point sets using only $2^k - 1$ possible labels for which every separability preserving projection also keeps a_1 strictly linearly separable after projection. The idea is to use the properties a_2, \dots, a_k to sufficiently restrict the direction of a separability preserving projection to make it impossible for this projection to eliminate the linear separability of a_1 . A simple example for $d = k = 2$ is shown in Figure 3.



■ **Figure 3** A point set with 5 points and 2 properties: a_1 (color) and a_2 (shape). To keep a_2 linearly separable after projection, the projection vector w should be nearly vertical, but then a_1 will also remain linearly separable.

► **Lemma 4.** *For all $k > 1$ and $d \geq k$, there exist point sets P in \mathbb{R}^d with properties a_1, \dots, a_k using $2^k - 1$ labels such that any separability preserving projection along a unit vector w also keeps a_1 strictly linearly separable after projection.*

We now assume that all 2^k labels are used in P . Note that this assumption directly implies that $d \geq k$: take any set of k separating hyperplanes H_1, \dots, H_k for the k properties and consider the arrangement formed by the hyperplanes in \mathbb{R}^d . Clearly, all points in the same cell of the arrangement must have the same label. However, it is well-known that it is



■ **Figure 4** Theorem 5 in 2D: 4 points are needed to construct two intersecting convex hulls.

not possible to create 2^k cells in \mathbb{R}^d with only k hyperplanes if $d < k$. This has also interesting implications for the case when $d = k$: if we apply a separation preserving projection to P , then a_1 cannot be linearly separable in P' , since P' is embedded in \mathbb{R}^{k-1} .

We now show that, if $d \geq k$, then there always exists a separation preserving projection that eliminates the strict linear separability of a_1 (see Figure 4). Our proof uses Kirchner's theorem [16]. Below we restate this theorem in our own notation. We also include our own proof, since the construction in the proof is necessary for efficient computation of our result.

► **Theorem 5** ([16]). *Let P and Q be two points sets in \mathbb{R}^d such that $CH(P) \cap CH(Q) \neq \emptyset$. Then there exist subsets $P^* \subseteq P$ and $Q^* \subseteq Q$ such that $CH(P^*) \cap CH(Q^*) \neq \emptyset$ and $|P^*| + |Q^*| = d + 2$.*

Proof. Let $|P| = n$ and $|Q| = m$. We show that, if $n + m \geq d + 3$, then we can remove one of the points from either P or Q . Pick a point $x \in CH(P) \cap CH(Q)$. By definition, we can find coefficients $\lambda_1, \dots, \lambda_n \geq 0$ and $\mu_1, \dots, \mu_m \geq 0$ such that $\sum_{i=1}^n \lambda_i p_i = x = \sum_{j=1}^m \mu_j q_j$, $\sum_{i=1}^n \lambda_i = 1$, and $\sum_{j=1}^m \mu_j = 1$. If any of these coefficients is zero, then we can remove the corresponding point whilst keeping x in the intersection of the two convex hulls. Otherwise, we find nonzero coefficients a_1, \dots, a_n and b_1, \dots, b_m such that $\sum_{i=1}^n a_i p_i = \sum_{j=1}^m b_j q_j$, $\sum_{i=1}^n a_i = 0$, and $\sum_{j=1}^m b_j = 0$. As this is a linear system with $d + 2$ constraints and $n + m \geq d + 3$ variables, there must exist a set of nonzero coefficients that satisfy these constraints. Let $\rho_\lambda = \min\{\lambda_i/a_i \mid a_i > 0\}$, $\rho_\mu = \min\{\mu_j/b_j \mid b_j > 0\}$, and $\rho = \min(\rho_\lambda, \rho_\mu)$. Now consider the new coefficients $\lambda'_i = \lambda_i - \rho a_i$ and $\mu'_j = \mu_j - \rho b_j$. By construction we have that $\lambda'_i \geq 0$ for $1 \leq i \leq n$, $\mu'_j \geq 0$ for $1 \leq j \leq m$, $\sum_i \lambda'_i = \sum_j \mu'_j = 1$, and $\sum_i \lambda'_i p_i = \sum_j \mu'_j q_j = x'$. Additionally, one of the new coefficients is zero, and we can remove the corresponding point. We can repeat this process until $n + m = d + 2$. ◀

The following proof constructs a suitable projection vector using four main steps:

1. We project the points orthogonally onto the linear subspace A spanned by the normals of the separating hyperplanes H_2, \dots, H_k .
2. We argue that, since P uses all 2^k labels, a_1 is not linearly separable in A .
3. We find a small subset of points P^* for which a_1 is not linearly separable in A .
4. We construct a separation preserving projection that maps all points in P^* to an affine transformation of A . As a result, a_1 is not strictly linearly separable after projection.

We assume that the points in P , along with the chosen separating hyperplanes, are in *general position*. Specifically, we assume that any set of d vectors, where each vector is either a distinct difference vector of two points in P or the normal vector of one of the separating hyperplanes, is linearly independent. Note that, since all properties are initially strictly linearly separable, it is always possible to perturb the separating hyperplanes to ensure general position, assuming that P is also in general position.

► **Theorem 6.** *If P is a point set in \mathbb{R}^d in general position with $k \leq d$ strictly linearly separable properties a_1, \dots, a_k using all 2^k labels, then there exists a separation preserving projection along a unit vector w that eliminates the strict linear separability of a_1 .*

Proof. We provide an explicit construction of the vector w . Let H_2, \dots, H_k be any separating hyperplanes (in general position with P) for each of the properties a_2, \dots, a_k in P , respectively. Let v_i be the normal of hyperplane H_i for $2 \leq i \leq k$, and let $A \subset \mathbb{R}^d$ be the $(k-1)$ -dimensional linear subspace spanned by v_2, \dots, v_k . Furthermore, let $H^* = \bigcap_{i=2}^k H_i$ be the $(d-k+1)$ -dimensional flat that is the intersection of the separating hyperplanes. Note that a projection along a vector w is separation preserving if and only if w is parallel to H^* . Let $T(p)$ be the result of an orthogonal projection of a point $p \in P$ onto A . For ease of argument, we also directly apply an affine transformation that maps H^* (which intersects A in one point by construction) to the origin, and maps v_2, \dots, v_k to the standard basis vectors of \mathbb{R}^{k-1} .

Now define $Q_- = \{T(p) \mid p \in P_-\}$ and $Q_+ = \{T(p) \mid p \in P_+\}$. By construction, since all labels are used by P , both Q_- and Q_+ must have a point in each orthant of \mathbb{R}^{k-1} . If a point set Q has a point in each orthant, then $CH(Q)$ must contain the origin; because if it does not, then there exists a vector v such that $(v \cdot q) > 0$ for all $q \in Q$. But there must exist a point $q^* \in Q$ whose sign for each coordinate is opposite from that of v (or zero), which means that $(v \cdot q^*) \leq 0$, a contradiction. Thus, both $CH(Q_-)$ and $CH(Q_+)$ contain the origin, and $CH(Q_-) \cap CH(Q_+) \neq \emptyset$. We now apply Theorem 5 to Q_- and Q_+ to obtain Q_-^* and Q_+^* consisting of $k+1$ points in total. Let $P^* \subseteq P$ be the corresponding set of original points that map to $Q_-^* \cup Q_+^*$. We can now construct w as follows. Pick a point $p^* \in P^*$, and let F_1 be the unique $(k-1)$ -dimensional flat that contains the remaining points in P^* . Let F_2 be the flat obtained by translating H^* to contain p^* . Since F_1 is $(k-1)$ -dimensional and F_2 is $(d-k+1)$ -dimensional, $F_1 \cap F_2$ consists of a single point $r \in \mathbb{R}^d$ (assuming general position). The desired projection vector is now simply $w = r - p^*$ (normalized if necessary).

We finally show that the constructed vector w has the correct properties. First of all, w is parallel to H^* by construction, and hence the projection along w is separation preserving. Second, since $r \in F_1$ and p^* is projected to coincide with r (as $w = r - p^*$), all points in P^* will lie on the same $(k-1)$ -dimensional flat F_1' after projection. Also, since w is orthogonal to A , there exists an affine map from $Q_-^* \cup Q_+^*$ to P^* (after projection). Thus, we obtain that $CH(P_-') \cap CH(P_+') \neq \emptyset$; in particular, the convex hulls must intersect on F_1' . By Fact 1 this implies that a_1 is not strictly linearly separable after projection. ◀

The result of Theorem 6 has one shortcoming: the resulting projected point set P' is degenerate by construction and property a_1 may still be (non-strictly) linearly separable after projection. This is simply an artifact of the proof and can be avoided by slightly perturbing the projection vector w . The following lemma can be used to remedy this shortcoming. Here we again assume that, before projection, the point set P and the separating hyperplanes are in general position, and hence the only degeneracy in P' is the one introduced by construction.

► **Lemma 7.** *Let P and Q be two point sets in \mathbb{R}^d in general position and let P' and Q' be the point sets obtained by projecting P and Q along a vector w , respectively. If $CH(P') \cap CH(Q') \neq \emptyset$, then we can perturb w to obtain projections P'' and Q'' such that P'' and Q'' are not linearly separable and $P'' \cup Q''$ is in general position.*

Proof. Let $P = \{p_1, \dots, p_n\}$ and $Q = \{q_1, \dots, q_m\}$, and similarly $P' = \{p'_1, \dots, p'_n\}$ and $Q' = \{q'_1, \dots, q'_m\}$. We may assume that $m+n \geq d+1$, for otherwise P and Q do not really span \mathbb{R}^d . Since $CH(P') \cap CH(Q') \neq \emptyset$, there exist coefficients $\lambda_i \geq 0$ ($1 \leq i \leq n$) and $\mu_j \geq 0$ ($1 \leq j \leq m$) such that $\sum_i \lambda_i = 1$, $\sum_j \mu_j = 1$, and $\sum_i \lambda_i p'_i = \sum_j \mu_j q'_j$. We can ignore some

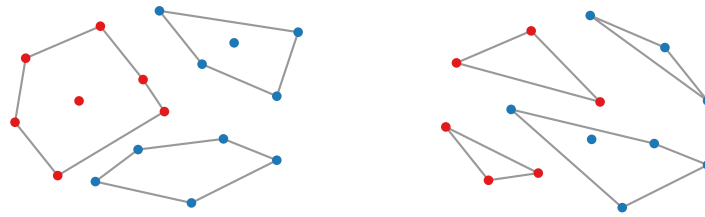
points with a zero coefficient so that we have exactly $d + 1$ points left, and we assume in the remainder of this proof that $m + n = d + 1$. Now assume w.l.o.g. that $\lambda_1 > 0$. We use the remaining points $(P' \cup Q') \setminus \{p'_1\}$ to set up a barycentric coordinate system for the points in $P' \cup Q'$. This has the advantage that only the coordinates of p_1 are affected when changing the projection vector w . Next, we slightly perturb the coefficients to obtain $\lambda'_i > 0$ ($1 \leq i \leq n$), $\mu'_j > 0$ ($1 \leq j \leq m$) with $\lambda'_1 = \lambda_1$, $\sum_i \lambda'_i = 1$ and $\sum_j \mu'_j = 1$ (this is clearly possible). There then exist a vector v (in barycentric coordinates) and $\epsilon > 0$ (ϵ can be arbitrarily small by scaling the perturbation of the coefficients) such that $\epsilon v + \sum_i \lambda'_i p'_i = \sum_j \mu'_j q'_j$. Now consider the point p_1^\perp which has the same barycentric coordinates as p'_1 , but then with the barycentric coordinate system defined by $(P \cup Q) \setminus \{p_1\}$. Then, by Observation 2, we must have that $p_1 - p_1^\perp = \alpha w$ for some constant $\alpha \neq 0$. Now we perturb p_1^\perp to p^* such that p^* has the same barycentric coordinates as $p'_1 + (\epsilon/\lambda_1)v$, but then again with the barycentric coordinate system defined by $(P \cup Q) \setminus \{p_1\}$. Additionally, we perturb w to $w' = p_1 - p^*$. Let $P'' = \{p''_1, \dots, p''_n\}$ and $Q'' = \{q''_1, \dots, q''_m\}$ be the point sets obtained by projecting P and Q along w' . We then have by construction that $\sum_i \lambda'_i p''_i = \sum_j \mu'_j q''_j$. Now assume for the sake of contradiction that P'' and Q'' are linearly separable by a hyperplane H . Then H must contain $CH(P'') \cap CH(Q'')$ and, consequently, all points that have a nonzero coefficient for the convex combination of a point $x \in CH(P'') \cap CH(Q'')$ (since all points of either P'' or Q'' lie on the same side of H). By construction there are $d + 1$ of these points in $P'' \cup Q''$. Since H is $(d - 2)$ -dimensional and we performed only a single projection, this also implies that there were $d + 1$ points on a $(d - 1)$ -dimensional hyperplane in $P \cup Q$. This contradicts the assumption that $P \cup Q$ is in general position. Finally, since $CH(P'')$ and $CH(Q'')$ are not interior disjoint by Fact 1, this property cannot be broken by slightly perturbing the projection vector w' . Thus, we can also ensure that $P'' \cup Q''$ is in general position. ◀

Computation. The proof of Theorem 6 is constructive and implies an efficient algorithm to compute the desired projection. Most steps in the construction involve simple linear algebra operations, like projections and intersecting flats (Gaussian elimination), which can easily be computed in polynomial time. The only nontrivial computational step is the application of Theorem 5, for which the proof is also constructive. If a point $x \in CH(P) \cap CH(Q)$ is given along with the coefficients for the convex combination, then we can simply obtain P^* and Q^* by repeatedly solving a linear system of equations and eliminating a point. Note that the linear system needs to involve only $d + 3$ points (arbitrarily chosen), so the linear system of equations can be solved in $O(d^3)$ time, and we can eliminate a point and update the coefficients in the same amount of time. Thus, we can compute P^* and Q^* in $O(nd^3)$ time, where $n = |P| + |Q|$ (similar arguments were used in [20]). If we are not given a point in $x \in CH(P) \cap CH(Q)$ along with the coefficients for the convex combination, then this must be computed first. This can be computed efficiently using linear programming.

The proof of Theorem 6 suggests how to check, if P does not use all 2^k labels, if there exists a separability preserving projection that eliminates the linear separability of a_1 : If we can find a set of separating hyperplanes H_2, \dots, H_k such that $CH(P_-)$ and $CH(P_+)$ intersect *after* projecting them orthogonally onto the space spanned by the normals of H_2, \dots, H_k , then the remainder of the proof holds. However, finding such suitable separating hyperplanes might be computationally hard in general.

3 Generalized separability

In this section we consider a generalization of linear separability for classification. One approach to achieve more complicated classification boundaries is to use clustering: the label of a point is determined by the label of the “nearest” cluster. If we use more than one



■ **Figure 5** Left: two point sets P (red) and Q (blue) that are $(1, 2)$ -separable, but not linearly separable. Right: two point sets that are $(2, 2)$ -separable, but not $(1, x)$ -separable for any value of x .

cluster per class, then the resulting classification is more expressive than classification by linear separation. This approach is also strongly related to nearest-neighbor classification, another common machine learning technique: the points decompose the space into convex subsets, each of which is associated with exactly one point; given enough clusters, we can thus exactly capture this behavior. But even with few clusters (convex sets), it may be possible to reasonably approximate the decomposition by using a single cluster to capture the same of many points with the same label. Hence, Our generalized definition of separability is inspired by such clustering-based classifications, with convex sets modeling the clusters.

Let P and Q be two point sets in \mathbb{R}^d . We say that P and Q are (b, c) -separable if there exist b convex sets S_1, \dots, S_b and c convex sets T_1, \dots, T_c such that for every point $p \in P$ we have that $p \in S = \bigcup_i S_i$, for every point $q \in Q$ we have that $q \in T = \bigcup_j T_j$, and that $S \cap T = \emptyset$ (see Figure 5). We can assume that $b \leq c$. Furthermore, we generally assume w.l.o.g. that any convex set S_i is the convex hull of its contained points. It is easy to see that linear separability and $(1, 1)$ -separability are equivalent.

Given a point set P along with k properties a_1, \dots, a_k , the goal is now to compute a separation preserving projection to a point set P' such that a_1 is not (b, c) -separable in P' . We again assume that all k properties are strictly linearly separable in P . To achieve this goal, we may need to project along multiple vectors w_1, \dots, w_r . As mentioned in Section 1, we assume that $\{w_j\}_{j=1}^r$ form an orthonormal system and that we can compute the projected points as $p'_i = p_i - \sum_{j=1}^r (w_j \cdot p_i) w_j$.

To extend Theorem 6 to (b, c) -separability, recall the four main steps of the proof described before Theorem 6. Step 3 is the most important. If a_1 was not linearly separable in A , then not even multiple separation preserving projections can eliminate the linear separability of a_1 . In that sense, A is the “worst we can do” with separation preserving projections. Step 3 is actually exploiting a Helly-type property [24] for linear separability: If two sets of points P and Q are not linearly separable, then there exist small subsets $P^* \subseteq P$ and $Q^* \subseteq Q$ such that P^* and Q^* are not linearly separable (Theorem 5). Hence, if we use a different type of separability that also has a Helly-type property, then we may be able to use the same approach as for linear separability. Generally speaking, let $F(P, Q)$ be a predicate that determines if point sets $P, Q \subseteq \mathbb{R}^d$ are “separable” (for some arbitrary definition of separable)². If, in the case that $F(P, Q)$ does not hold, there exist small (bounded by a constant) subsets $P^* \subseteq P$ and $Q^* \subseteq Q$ such that $F(P^*, Q^*)$ also does not hold, then F has the *Helly-type property*. The worst-case size of $|P^*| + |Q^*|$ often depends on the number of dimensions d of P and Q , and is referred to as the *Helly number* $m_F(d)$ of F . For technical reasons, we will require the following three natural conditions on F :

² We assume that F is defined independently from the dimensionality of P and Q (like (b, c) -separability). We do require that P and Q are embedded in the same space.

56:10 Obstructing Classification via Projection

1. If $F(P, Q)$ does not hold, then $F(P', Q')$ does not hold, where P' and Q' are obtained by projecting P and Q along a single unit vector, respectively.
2. If $P' \subseteq P$ and $Q' \subseteq Q$, then $F(P, Q)$ implies $F(P', Q')$.
3. If \mathcal{A} is an affine map, then $F(P, Q)$ holds if and only if $F(\mathcal{A}(P), \mathcal{A}(Q))$ holds.

We call a separation predicate F *well-behaved* if it satisfies these conditions. It is easy to see that (b, c) -separability is well-behaved. For Condition 1, note that any collection of convex sets for P' and Q' can easily be extended along the projection vector for P and Q without introducing an overlap between S and T . Condition 2 also holds, since we can simply use the same covering sets. Finally, Condition 3 holds since affine transformations preserve convexity. We summarize this generalization in the following generic theorem.

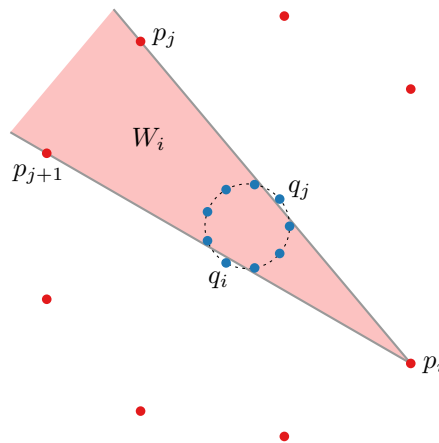
► **Theorem 8.** *Let P be a point set in \mathbb{R}^d with k ($d \geq k$) properties a_1, \dots, a_k and let F be a well-behaved separation predicate in \mathbb{R}^d . Either we can use at most $\min(m_F(k-1)-k, d-k+1)$ separation preserving projections to eliminate $F(P_-, P_+)$, or this cannot be achieved with any number of separation preserving projections.*

Proof. Following the proof of Theorem 6, we first orthogonally project the points in P onto the $(k-1)$ -dimensional linear subspace A that is spanned by the normals v_2, \dots, v_k of the separating hyperplanes H_2, \dots, H_k of the properties a_2, \dots, a_k . Let $T(p)$ be the resulting projected point for a point $p \in P$. Now define $Q_- = \{T(p) \mid p \in P_-\}$ and $Q_+ = \{T(p) \mid p \in P_+\}$. If $F(Q_-, Q_+)$ holds, then no sequence of separation preserving projections can eliminate the separability (as defined by F) of a_1 , due to Condition 1 of a well-behaved separation predicate. Otherwise, we can find $Q_-^* \subseteq Q_-$ and $Q_+^* \subseteq Q_+$ such that $F(Q_-^*, Q_+^*)$ does not hold, and $|Q_-^*| + |Q_+^*| \leq m_F(k-1)$. Let $P^* \subseteq P$ be the set of original points that map to $Q_-^* \cup Q_+^*$. The points in P^* span a linear subspace B . Next, we construct an orthonormal basis $\{w_j\}_{j=1}^r$ for the set of vectors in B that are orthogonal to A (orthogonal to v_2, \dots, v_k). Since B has at most $m_F(k-1) - 1$ dimensions, and A has $k-1$ dimensions, we conclude that the orthonormal basis contains $r \leq m_F(k-1) - 1 - (k-1) = m_F(k-1) - k$ vectors. We then choose to project P along the vectors w_1, \dots, w_r . Since every w_j for $1 \leq j \leq r$ is orthogonal to A , these projections are all separation preserving. Furthermore, since we eliminate all vectors orthogonal to A from B , there exists an affine map from $Q_-^* \cup Q_+^*$ to P^* after projection. By using Condition 2 and Condition 3 of a well-behaved separation predicate, we can then conclude that $F(P'_-, P'_+)$ does not hold. Alternatively, we can simply project P to A , which requires $d - k + 1$ separation preserving projections. Hence, we need at most $\min(m_F(k-1) - k, d - k + 1)$ projections. ◀

We now focus on (b, c) -separability for different values of b and c . Unfortunately, not every form of (b, c) -separability has the Helly-type property.

► **Lemma 9.** *In $d \geq 2$ dimensions, $(1, 2)$ -separability does not have the Helly-type property.*

Proof. We prove the statement for $d = 2$, which automatically implies it for $d > 2$. Consider a set of n points $P = \{p_1, \dots, p_n\}$ equally spaced on the unit circle, where n is odd. For every point p_i we can define a wedge W_i formed between the rays from p_i to the two opposite points on the circle (which are well defined, since n is odd). By the Central Angle Theorem, the angle of this wedge is $\frac{\pi}{n}$. Furthermore, the distance of the rays to the origin is exactly $\sin(\frac{\pi}{n})$. Now, for some $\epsilon > 0$ and for each point p_i , we add a point q_i on the circle centered at the origin with radius $\sin(\frac{\pi}{n}) + \epsilon$, such that q_i lies outside of W_i to the left (counterclockwise). By construction there will also be a point q_j to the right of W_i , added by the point p_j that is the opposite point of p_i on the right (clockwise) side. We choose ϵ small enough such that any wedge W_i contains exactly $n - 2$ points from $Q = \{q_1, \dots, q_n\}$, having one point of Q outside of W_i on each side (see Figure 6).



■ **Figure 6** The construction for Lemma 9 with P in red and Q in blue.

Assume for the sake of contradiction that P and Q are $(1, 2)$ -separable. Since $Q \subset CH(P)$, we must cover Q with one set, and hence $S_1 = CH(Q)$. Now consider $P_1 = T_1 \cap P$ and $P_2 = T_2 \cap P$. Since the line segments between a point $p_i \in P_1$ and its opposite points p_j and p_{j+1} intersect $CH(Q)$, we get that p_j and p_{j+1} must both be in P_2 . We can repeat this argument for all points p_i to conclude that all pairs of consecutive points of P must be in the same set (P_1 or P_2). Since not all points in P can belong to the same set ($Q \subset CH(P)$), we obtain a contradiction. Thus, P and Q are not $(1, 2)$ -separable.

Now consider removing a single point p_i from P , and consider the line ℓ through the origin and p_i . The line ℓ splits $P \setminus \{p_i\}$ into two sets P_1 and P_2 . It is easy to see that, if we pick ϵ small enough, $CH(P_1)$ and $CH(P_2)$ do not intersect $CH(Q)$. Hence, $P \setminus \{p_i\}$ and Q are $(1, 2)$ -separable. If we remove a single point q_i from Q , then the line segment between p_i and one of its opposite points p_j does not intersect $CH(Q \setminus \{q_i\})$. We can again split P into P_1 and P_2 using the line ℓ through p_i and p_j (and shifted slightly towards the origin). Then it is again easy to see that, if we pick ϵ small enough, $CH(P_1)$ and $CH(P_2)$ do not intersect $CH(Q \setminus \{q_i\})$. Hence, P and $Q \setminus \{q_i\}$ are $(1, 2)$ -separable.

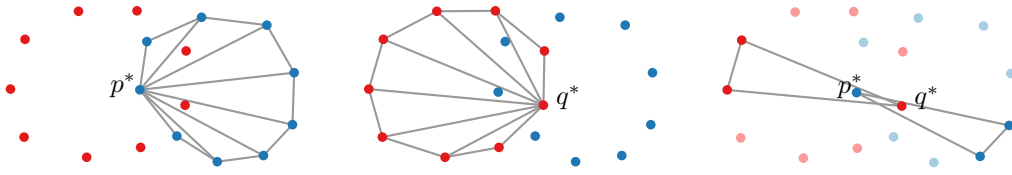
As a result, there exist no subsets of P and Q that are not $(1, 2)$ -separable. Thus, we get that the Helly number for $(1, 2)$ -separability is at least $|P| + |Q| = 2n$, and hence $(1, 2)$ -separability does not have the Helly-type property. ◀

Hence we cannot apply Theorem 8 to eliminate $(1, 2)$ -separability of a_1 in few separation preserving projections, if possible at all. However, this does not mean that it is not possible to provide this guarantee using different arguments. Nonetheless, we can use a similar construction as in the proof of Lemma 9 (using many more dimensions) to show that many separation preserving projections are needed to eliminate $(1, 2)$ -separability for a_1 (as many projections as needed to reach the 2-dimensional construction in the proof of Lemma 9).

Next, we consider $(1, \infty)$ -separability. This means that one of the point sets, say P , must be covered with one convex set, but we can use arbitrarily many convex sets to cover Q . Equivalently, P and Q are $(1, \infty)$ -separable if $CH(P) \cap Q = \emptyset$ or $P \cap CH(Q) = \emptyset$.

► **Lemma 10.** *In $d \geq 1$ dimensions, $(1, \infty)$ -separability has the Helly-type property with Helly number $2d + 2$.*

56:12 Obstructing Classification via Projection



■ **Figure 7** Lemma 10: constructing a small point set that is not $(1, \infty)$ -separable.

Proof. Let P and Q be point sets in \mathbb{R}^d such that P and Q are not $(1, \infty)$ -separable. Then there must be a point $p^* \in CH(Q)$ and a point $q^* \in CH(P)$. We can construct a star triangulation $\mathcal{T}(P)$ of $CH(P)$ with p^* as center (that is, all d -dimensional simplices have p^* as a vertex) and a star triangulation $\mathcal{T}(Q)$ of $CH(Q)$ with q^* as center (see Figure 7). We identify the unique simplex $\sigma_P \in \mathcal{T}(P)$ that contains q^* , and similarly the unique simplex $\sigma_Q \in \mathcal{T}(Q)$ that contains p^* . Now let $P^* \subseteq P$ be the vertices of σ_P and let $Q^* \subseteq Q$ be the vertices of σ_Q . Note that $p^* \in P^*$ and $q^* \in Q^*$. Then P^* and Q^* are not $(1, \infty)$ -separable, since $q^* \in CH(P^*) \cap Q^*$ and $p^* \in CH(Q^*) \cap P^*$. Finally, since a d -dimensional simplex contains $d + 1$ vertices, we obtain Helly number $2d + 2$. ◀

► **Corollary 11.** *Let P be a point set in \mathbb{R}^d with k ($d \geq k$) properties a_1, \dots, a_k . Either we can use at most $\min(k, d - k + 1)$ separation preserving projections to eliminate $(1, \infty)$ -separability of a_1 , or this cannot be achieved with any number of separation preserving projections.*

It may initially seem counter-intuitive that $(1, \infty)$ -separability has the Helly-type property (requiring only few projections to eliminate $(1, \infty)$ -separability), while the strictly stronger $(1, 2)$ -separability does not have the Helly-type property (and may require many projections to eliminate $(1, 2)$ -separability). Note however that Theorem 8 includes the clause that it simply may not be possible to eliminate separability of a_1 via any number of separation preserving projections. This case occurs more often with $(1, \infty)$ -separability than with $(1, 2)$ -separability, which explains why we can provide better guarantees on the number of projections for a strictly weaker separability condition.

We finally briefly consider $(2, \infty)$ -separability in \mathbb{R}^2 . Two point sets P and Q are not $(2, \infty)$ -separable in \mathbb{R}^2 if we need at least three convex sets disjoint from Q to cover P (and vice versa). This implies that $CH(P)$ must contain at least 3 points of Q ; if not, then we can draw a single line through all points in $Q \cap CH(P)$ to separate P into P_1 and P_2 , and $CH(P_1)$ and $CH(P_2)$ both cover P and are disjoint from Q . More generally, assume that we can cover P with two sets $CH(P_1)$ and $CH(P_2)$ that are disjoint from Q , and let ℓ be a line that separates $CH(P_1)$ and $CH(P_2)$ (Fact 1). Now consider the set of all triangles \mathcal{T}_P that are formed by three points of P such that a point of Q is contained in the triangle. We must have that ℓ transverses (intersects) all triangles in \mathcal{T}_P , otherwise the triangle is contained in P_1 or P_2 , and hence there is a point of Q in either $CH(P_1)$ or $CH(P_2)$. Furthermore, if there is point $q \in Q$ contained in, say, $CH(P_1)$, then there is also a triangle $\Delta \in \mathcal{T}_P$ in P_1 (Carathéodory's theorem), and hence ℓ does not intersect all triangles in \mathcal{T}_P . Thus, P and Q are $(2, \infty)$ -separable (assuming we cover P with 2 convex sets) if and only if there exists a line ℓ that transverses \mathcal{T}_P . As a result, if we can show a Helly-type property for line transversals of triangles, then we also obtain a Helly-type property for $(2, \infty)$ -separability. Unfortunately, there is no Helly-type property for line transversals of general sets of triangles [17]. We leave it as an open question to determine if there exists a Helly-type property for line transversals of these special sets of triangles \mathcal{T}_P .

4 Conclusion

We studied the use of projections for obstructing classification of high-dimensional Euclidean point data. Our results show that, if not all possible labels are present in the data, then it may not be possible to eliminate the linear separability of one property while preserving it for the other properties. This is not surprising if a property that we aim to keep is strongly correlated with the property we aim to hide. Nonetheless, one should be aware of this effect when employing projections in practice. When going beyond linear separability, we see that the number of projections required to hide a property increases significantly in theory, and we expect a similar effect when using, for example, neural networks for classification in practice. In other words, projecting a dataset once (or few times) may not be sufficient to hide a property from a smart classifier. Projection, as a linear transformation, can however be effective in eliminating certain linear relations in the data.

One potential direction of future work is to consider other separability predicates for labeled point sets, beyond linear separability and (b, c) -separability. Are there other types of separability that also have the Helly-type property used in Theorem 8? Or is there another way to show that few projections suffice to eliminate the separability of one of the properties? There are many other types of separability (for example, via boxes or spheres) for which this can be evaluated.

In this paper we focused on eliminating bias based on one property (such as gender). Intersectionality posits that discrimination due to multiple properties should be considered in a holistic manner, instead of one property at a time. In fact, any one property might not be a cause for discrimination, but their combination is. The following challenge arises: say we used projection successfully to eliminate the linear separability of gender. However, if we now restrict the data to one particular sub-class, for example black people, then the linear separability of gender might still be preserved within this subclass and hence discrimination against black women can still be possible. Under which conditions is it possible to eliminate the linear separability of one property not only in the full data, but also in specific (or all) subclasses? We leave this question as an open problem.

References

- 1 Mohsen Abbasi, Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. Fairness in representation: quantifying stereotyping as a representational harm. In *Proc. SIAM International Conference on Data Mining*, pages 801–809, 2019. doi:10.1137/1.9781611975673.90.
- 2 Alexander Amini, Ava P. Soleimany, Wilko Schwarting, Sangeeta N. Bhatia, and Daniela Rus. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proc. AAAI/ACM Conference on AI, Ethics, and Society*, pages 289–295, 2019. doi:10.1145/3306618.3314243.
- 3 Richard Berk, Hoda Heidari, Shahin Jabbari, Matthew Joseph, Michael J. Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. A convex framework for fair regression. *arXiv:1706.02409*, 2017.
- 4 Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.
- 5 Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 4349–4357, 2016.

- 6 Shikha Bordia and Samuel R. Bowman. Identifying and reducing gender bias in word-level language models. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 7–15, 2019. doi:10.18653/v1/n19-3002.
- 7 Marc E. Brunet, Colleen A. Houlihan, Ashton Anderson, and Richard S. Zemel. Understanding the origins of bias in word embeddings. In *Proc. 36th International Conference on Machine Learning*, volume 97, pages 803–811, 2019.
- 8 Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017. doi:10.1126/science.aal4230.
- 9 Sunipa Dev, Tao Li, Jeff M. Phillips, and Vivek Srikumar. On measuring and mitigating biased inferences of word embeddings. In *Proc. AAAI Conference on Artificial Intelligence*, pages 7659–7666, 2020.
- 10 Sunipa Dev and Jeff Phillips. Attenuating bias in word vectors. In *Proc. Machine Learning Research*, pages 879–887, 2019.
- 11 Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *arXiv*, 2016. arXiv:1511.05897.
- 12 Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proc. 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 259–268, 2015.
- 13 Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 3315–3323, 2016.
- 14 Yuzi He, Keith Burghardt, and Kristina Lerman. A geometric solution to fair representations. In *Proc. AAAI/ACM Conference on AI, Ethics, and Society*, page 279–285, 2020. doi:10.1145/3375627.3375864.
- 15 Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33:1–33, 2011. doi:10.1007/s10115-011-0463-8.
- 16 Paul Kirchberger. Über Tchebychefsche Annäherungsmethoden. *Mathematische Annalen*, 57:509–540, 1903. doi:10.1007/BF01445182.
- 17 Ted Lewis. Two counterexamples concerning transversals for convex subsets of the plane. *Geometriae Dedicata*, 9:461–465, 1980. doi:10.1007/BF00181561.
- 18 David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations. In *Proc. 35th International Conference on Machine Learning*, pages 3384–3393, 2018.
- 19 Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *arXiv*, 2019. arXiv:1908.09635.
- 20 Frédéric Meunier, Wolfgang Mulzer, Pauline Sarrabezolles, and Yannik Stein. The rainbow at the end of the line - A PPAD formulation of the colorful Carathéodory theorem with applications. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms*, pages 1342–1351, 2017. doi:10.1137/1.9781611974782.87.
- 21 Yingjie Tian Naiyang Deng and Chunhua Zhang. *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*. CRC Press, 2012.
- 22 Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, 2020.
- 23 Jennifer L. Skeem and Christopher T. Lowenkamp. Risk, race, and recidivism: Predictive bias and disparate impact. *Criminology*, 54(4):680–712, 2016. doi:10.1111/1745-9125.12123.
- 24 Rephael Wenger. Helly-type theorems and geometric transversals. In Jacob E. Goodman and Joseph O’Rourke, editors, *Handbook of Discrete and Computational Geometry, second edition*, pages 73–96. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.ch4.

- 25 Muhammad Bilal Zafar, Isabel Valera, Manuel G. Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *Proc. 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 962–970, 2017.
- 26 Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *Proc. 30th International Conference on Machine Learning*, pages 325–333, 2013.
- 27 Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proc. AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340, 2018. doi:10.1145/3278721.3278779.
- 28 Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai Wei Chang. Learning gender-neutral word embeddings. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 4847–4853, 2018.

A Maximizing inseparability

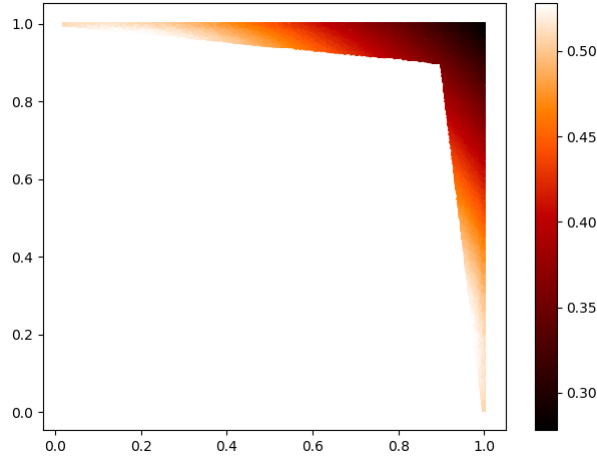
In this section we consider the problem of not only eliminating the linear separability of a_1 , but additionally to maximize the “linear inseparability” (or overlap) of a_1 after projection. For that we need to define the overlap between two point sets P and Q . For a unit vector v , consider the intervals $I_P(v) = CH(\{v \cdot p_i \mid p_i \in P\})$ and $I_Q(v) = CH(\{v \cdot q_i \mid q_i \in Q\})$. We can then define the overlap between P and Q along v as the length of $I_P(v) \cap I_Q(v)$. Alternatively, we can define the overlap along v with the cost function used by soft-margin SVMs, which is designed for data that is not linearly separable (see [4] for more details). The overlap between two point sets P and Q is then defined as the minimum overlap over all (unit) vectors v . More precisely, for a given (projected) point set P , along with (implicit) property a_1 , we use the function $g(P, v)$ to describe the overlap of a_1 along the vector v , and we refer to g as the *overlap function*. The overlap of a_1 is then defined as $\min_v g(P, v)$. Our goal is to find the projection that maximizes this overlap after projection. More precisely, if we use $P' = \pi_w(P)$ to denote the projection of a point set P along the unit vector w , then the goal is to maximize the function $f(P, w) = \min_v g(\pi_w(P), v)$ over all separability/separation preserving projection vectors w . We consider the following two overlap functions $g(P, v)$ (although other options are possible):

Interval. For a point set P and unit vector v , let $I_- = CH(\{v \cdot p_i \mid p_i \in P_-\})$ and $I_+ = CH(\{v \cdot p_i \mid p_i \in P_+\})$. Then $g_{\text{int}}(P, v) = |I_- \cap I_+|$.

SVM. The goal of the soft-margin SVM optimization is to minimize $g_{\text{svm}}(P, v) = \lambda \|v\|^2 + \frac{1}{n} \sum_{i=1}^n \max(0, 1 - a_1(p_i)(v \cdot p_i - b))$. Note that g_{svm} also requires a parameter $b \in \mathbb{R}$, but we will often omit that dependence (we can assume that the overlap function minimizes over all $b \in \mathbb{R}$). Furthermore, $\lambda > 0$ is a parameter that can be set for g_{svm} . Finally, note that v does not need to be a unit vector.

We first consider the variant of the problem that aims to find the optimal separability preserving projection. The vector v that minimizes $g(P, v)$ for a given point set is typically computed using convex programming (in particular for SVMs, see [21]). Note that convex programming heavily relies on the fact that there exists only one local optimum (which hence must be the global optimum). We show that, for the problem of finding the optimal separability preserving projection, there may be multiple local optima for $f(P, w)$. This eliminates the hope of finding a convex programming formulation for this problem.

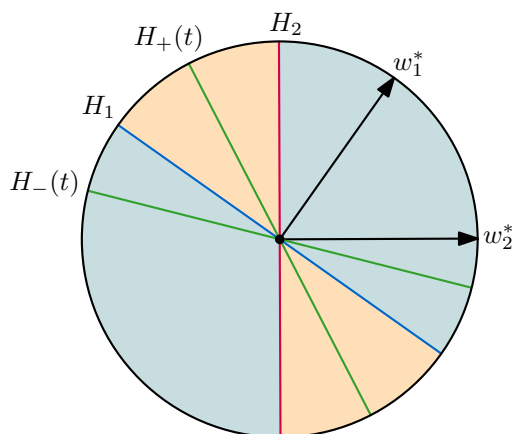
► **Theorem 12.** *There exists a point set P in \mathbb{R}^3 with 2 properties a_1, a_2 such that $f(P, w)$ with $g = g_{\text{svm}}$ has two local maxima when restricted to all separability preserving projection vectors w .*



■ **Figure 8** Illustration for Theorem 12: The domain for projections (x, y) with $\epsilon = 0.2$. Higher values in the overlap function are indicated with lighter colors. We can see two distinct local maxima.

Proof. The set P mostly consists of the vertices of a unit cube with side lengths 2 centered at the origin. We also add an extra point $p^* = (1 - \epsilon, 1 - \epsilon, 1)$ for some $\epsilon > 0$ (a point slightly moved inward from the point $p_8 = (1, 1, 1)$). Thus, P consists of nine points $\{p_1, \dots, p_8, p^*\}$. For property a_1 we choose that $a_1(p) = z(p)$ for all $p = (x(p), y(p), z(p)) \in P$. For property a_2 we have that $a_2(p) = a_1(p)$ for all $p \in P \setminus \{p_8\}$, and $a_2(p_8) = -1$. Now we limit and encode the space of possible projection vectors w to \mathbb{R}^2 as follows. For $w = (x(w), y(w), z(w))$ with $z(w) = 0$ it is clear that a projection along w will keep a_1 linearly separated, so we may encode all possible projections as $(x, y) = (x(w)/z(w), y(w)/z(w))$. Now consider the effect of using a projection (x, y) on P : we may assume that the x- and y-coordinates of points $p \in P$ with $z(p) = 1$ do not change and that for the other points we obtain a shifted square: $(x(p'), y(p')) = (x(p) + 2x, y(p) + 2y)$ for all $p \in P$ with $z(p) = -1$. Let $p_1 = (-1, -1, -1)$ such that $p'_1 = (-1 + 2x, -1 + 2y)$. Furthermore, let A consist of the projections of all points $p \in P$ with $a_1(p) = 1$, and let B consist of the projections of all points $p \in P$ with $a_2(p) = 1$. Note that A forms a square and B forms a square with one of the corners pushed inwards. By Fact 1, a projection (x, y) can only be separability preserving if $p'_1 \notin CH(B)$. By the same observation, a projection (x, y) with $x \geq 0$ and $y \geq 0$ preserves the linear separability of a_1 if $p'_1 \notin CH(A)$. Thus, we require that $p'_1 = (-1 + 2x, -1 + 2y) \in CH(A) \setminus CH(B)$. Note that $CH(A) \setminus CH(B)$ is a thin and nonconvex shape. The same thus holds for the domain of the projections (x, y) as shown in Figure 8, and hence the optimization problem is not convex. Furthermore, by evaluating the overlap function g_{svm} (using $\lambda = 10$) on this domain, we can see that there are two distinct local maxima: one close to $(0, 1)$ and one close to $(1, 0)$. ◀

Theorem 12 demonstrates that the constraint on projections to be separability preserving is generally not convex. We now consider the special case that we have only one property a_1 (hence no separability preserving constraint), and analyze if we can then efficiently maximize $f(P, w)$. For that, we first put a restriction on the overlap function $g(P, v)$. We say that $g(P, v)$ is *projectionable* if there exists a function $h: \mathbb{R}^n \times \mathbb{R}^d \rightarrow \mathbb{R}$ such that $g(P, v) = h(v \cdot P, v)$, where $v \cdot P = \{v \cdot p_i \mid p_i \in P\}$. In other words, g should only depend on P via the dot products of points in P with v . Note that both the Interval and SVM overlap functions are indeed projectionable. For projectionable overlap functions g we can redefine the optimization function f . In the following, let $v \perp w$ indicate that v and w are orthogonal.



■ **Figure 9** Illustration for Theorem 14 with $d = 2$: $H_+(t)$ can only intersect $R_{00} \cup R_{11}$ (orange, clipped to the unit disk) and $H_-(t)$ can only intersect $R_{01} \cup R_{10}$ (blue).

► **Lemma 13.** *If $g(P, v)$ is a projectionable overlap function, then $\max_w \min_v g(\pi_w(P), v) = \max_w \min_{v \perp w} g(P, v)$ for any point set $P \subset \mathbb{R}^d$.*

Proof. We will treat both w and v as a vector in \mathbb{R}^d . Since g is projectionable, there exists an equivalent function h that depends on v and the dot products between v and points in P . Now assume that $(v \cdot w) = 0$. Then we get

$$\begin{aligned} g(\pi_w(P), v) &= h(v \cdot \pi_w(P), v) \\ &= h(\{v \cdot (p_i - (w \cdot p_i)w) \mid p_i \in P\}, v) \\ &= h(\{v \cdot p_i \mid p_i \in P\}, v) \\ &= g(P, v). \end{aligned}$$

Since the vector v that minimizes $g(\pi_w(P), v)$ must be perpendicular to w , we obtain the desired equality. ◀

In the following we assume that the overlap function g is projectionable. Hence, by Lemma 13, we can rewrite f as $f(P, w) = \min_{v \perp w} g(P, v)$. This has the advantage that we can keep the point set P fixed while optimizing for w . We now aim to link properties of g to properties of f . As already discussed earlier, we can often find the vector v that minimizes $g(P, v)$ using convex programming. This implies that g has only one local minimum (for fixed P). We now use this fact to show that f has only one local maximum.

► **Theorem 14.** *If a function $g(P, v)$ has one local minimum for fixed P , then $f(P, w) = \min_{v \perp w} g(P, v)$ has one local maximum for fixed P .*

Proof. Note that $w \in \mathcal{S}^{d-1}$, where \mathcal{S}^{d-1} is the unit $(d - 1)$ -sphere, and $f(P, w) = f(P, -w)$, so we will treat w and $-w$ as equivalent. Similarly, if the local minimum of $g(P, v)$ is at $v = v^*$, then $v = -v^*$ may also be a local minimum, and together they will be counted as a single local minimum. For the sake of contradiction, assume that $f(P, w)$ has two distinct local maxima, one at $w = w_1^*$ and one at $w = w_2^*$ (and also at $w = -w_1^*$ and $w = -w_2^*$). We do not require that w_1^* and w_2^* are strict local maxima, but we do require that there exists no path $\gamma: [0, 1] \rightarrow \mathcal{S}^{d-1}$ with $\gamma(0) = w_1^*$ and $\gamma(1) = \pm w_2^*$ such that $f(P, \gamma(t)) \geq \min(f(P, \gamma(0)), f(P, \gamma(1)))$ for all $0 \leq t \leq 1$. Now consider the hyperplanes $H_1 = \{v \mid (v \cdot w_1^*) = 0\}$ and $H_2 = \{v \mid (v \cdot w_2^*) = 0\}$. Furthermore, let $\gamma_+: [0, 1] \rightarrow \mathcal{S}^{d-1}$ denote

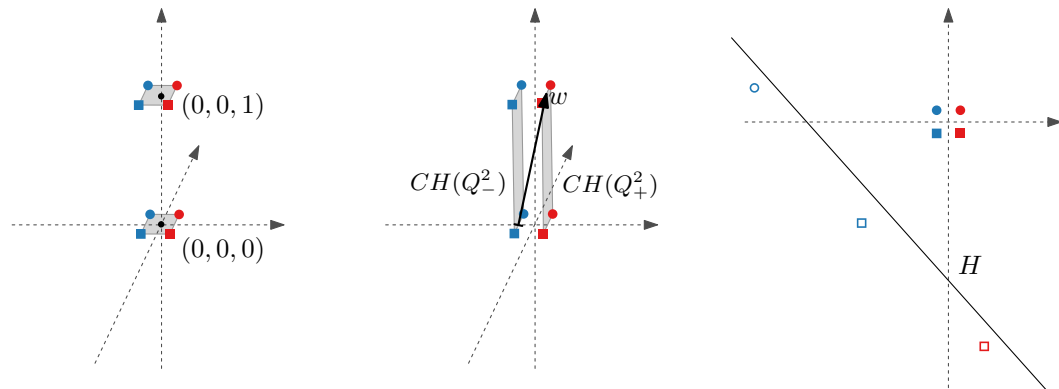
the shortest (hyper)spherical interpolation from w_1^* to w_2^* , and let $\gamma_- : [0, 1] \rightarrow \mathcal{S}^{d-1}$ denote the shortest (hyper)spherical interpolation from w_1^* to $-w_2^*$. Note that γ_- and γ_+ are unique, since both w_1^* to w_2^* lie on a great circle on \mathcal{S}^{d-1} and $w_1^* \neq -w_2^*$. The hyperplanes H_1 and H_2 split \mathbb{R}^d into four parts (each hyperplane cuts \mathbb{R}^d into two parts): R_{00}, R_{01}, R_{10} , and R_{11} . Now let $x^* = \min(f(P, w_1^*), f(P, w_2^*))$ and consider the sublevel set $S = \{v \mid g(P, v) < x^*\}$. By definition of f , H_1 and H_2 are disjoint from S . Now consider a vector $\gamma_+(t)$ for some $0 < t < 1$, and let $H_+(t) = \{v \mid (v \cdot \gamma_+(t)) = 0\}$ be the corresponding hyperplane. Similarly define $H_-(t)$ for $\gamma_-(t)$. It is easy to see that $H_+(t)$ intersects either $R_{01} \cup R_{10}$ or $R_{00} \cup R_{11}$, but not both, and $H_-(t)$ intersects only the other region (see Figure 9). By assumption, there exist values t_-^* and t_+^* such that $f(P, \gamma_-(t_-^*)) < x^*$ and $f(P, \gamma_+(t_+^*)) < x^*$. Thus, by the definition of f , there must be two non-opposite regions, say R_{00} and R_{01} , that contain a point in S . These points cannot be in the same connected component, as they are separated by either H_1 or H_2 . Thus, S has multiple (non-opposite) connected components, and hence $g(P, v)$ must have at least two local minima. This contradicts our assumption, and hence $f(P, w)$ can have at most one local maximum. ◀

Following Theorem 14, we can use a hill-climbing approach to find the optimal projection vector w , if there is only one property a_1 . This same approach can be applied to find the optimal separation preserving projection for k properties. In that case, the corresponding separating hyperplanes H_2, \dots, H_k each take away a degree of freedom, but otherwise do not bound the domain of w . More precisely, if there is only one property a_1 , then $w \in \mathcal{S}^{d-1}$, where \mathcal{S}^d is the unit d -sphere. If there are k properties, then $w \in \mathcal{S}^{d-1} \cap H_2 \cap \dots \cap H_k = \mathcal{S}^{d-k}$. This reduction in dimensionality of the domain of w does not affect the proof of Theorem 14.

B Omitted proofs

► **Lemma 4.** *For all $k > 1$ and $d \geq k$, there exist point sets P in \mathbb{R}^d with properties a_1, \dots, a_k using $2^k - 1$ labels such that any separability preserving projection along a unit vector w also keeps a_1 strictly linearly separable after projection.*

Proof. We first construct the point set P for arbitrary k and $d = k$. Consider the vertices of a $(k - 1)$ -dimensional hypercube C_ϵ with side length $\epsilon > 0$ centered at the origin, for which all nonzero coordinates lie in the first $k - 1$ dimensions of \mathbb{R}^d . For each vertex p of C_ϵ , set



■ **Figure 10** Illustration for Lemma 4 with $d = k = 3$ and properties fill (a_1), color (a_2), and shape (a_3). Left: Q consisting of two copies of C_ϵ . Middle: a separability preserving projection must be nearly orthogonal to the (x, y) -plane. Right: the flat H separating property a_1 .

the properties of p based on its coordinates (p^1, \dots, p^d) : $a_1(p) = 1$, and $a_i(p) = \text{sgn}(p^{i-1})$ for $2 \leq i \leq k$, where $\text{sgn}(x)$ is the sign function. Next, create a copy of C_ϵ (along with the assigned properties) and place it around the coordinate $(0, \dots, 0, 1)$ (see Figure 10 left). Let the resulting point set be Q , and consider projecting Q along a unit vector w . Let $w = (w^1, \dots, w^d)$ and assume w.l.o.g. that $|w^1| \geq |w^i|$ for all $2 \leq i < k$. If $|w^1| > \epsilon$, then there always exists a line ℓ parallel to w that intersects both $CH(Q_-^2)$ and $CH(Q_+^2)$. By Lemma 3 this would imply that a_2 is not strictly linearly separable after projection along w , so we may assume that $|w^1| \leq \epsilon$ for any separability preserving projection (see Figure 10 middle).

Now consider a $(k-2)$ -dimensional flat H with the following properties: (1) it is not parallel to one of the first $k-1$ axes, (2) it lies in the first $k-1$ dimensions of \mathbb{R}^d (the other coordinates are zero), and (3) the distance from the origin to H is 1 (see Figure 10 right). Consider the orthants of the $(k-1)$ -dimensional subspace A spanned by the first $k-1$ axes. Based on the labels of the vertices of C_ϵ , each orthant is associated with a label for the properties a_2, \dots, a_k . Due to Property (1), H intersects all the first $k-1$ axes, either at the positive or the negative half-axis. Since there is exactly one orthant bounded by only the non-intersected half-axes, H intersects exactly $2^{k-1} - 1$ orthants. We now construct P by extending Q with an additional point in each of the intersected orthants, such that H separates this point from the origin. The label of each such point p has $a_1(p) = -1$ and is otherwise determined by the orthant. As a result, P uses $2^k - 1$ different labels.

Let v be the normal of H in the $(k-1)$ -dimensional subspace A . The margin for P_- and P_+ along v is at least $1 - k\epsilon$ (rough bound). For any separability preserving projection along unit vector w , we have that $|(w \cdot v)| \leq \epsilon$. Now consider any point $p \in P$ and its projection $p' = p - (w \cdot p)w$. We have that $(p' \cdot v) = (p \cdot v) - (w \cdot p)(w \cdot v) = (p \cdot v) \pm O(\epsilon)$, where we use the fact that $(w \cdot p) = O(1)$. Thus, the margin for property a_1 can be reduced by at most $O(\epsilon)$ by the projection, and hence the projection keeps a_1 strictly linearly separable if we choose ϵ small enough.

If $d > k$, then we can construct a simplex with side lengths 1 in the last $d - k + 1$ dimensions, and place a copy of C_ϵ around each of its vertices (for $d = k$ this simplex is simply an edge, as used above). With this construction we can still enforce w.l.o.g. that $|w^1| \leq \epsilon$ for any separability preserving projection along unit vector w , and the rest of the argument follows. \blacktriangleleft

Online Domination: The Value of Getting to Know All Your Neighbors

Hovhannes A. Harutyunyan

Department of Computer Science and Software Engineering,
Concordia University, Montreal, Canada

Denis Pankratov

Department of Computer Science and Software Engineering,
Concordia University, Montreal, Canada

Jesse Racicot

Department of Computer Science and Software Engineering,
Concordia University, Montreal, Canada

Abstract

We study the dominating set problem in an online setting. An algorithm is required to guarantee competitiveness against an adversary that reveals the input graph one node at a time. When a node is revealed, the algorithm learns about the entire neighborhood of the node (including those nodes that have not yet been revealed). Furthermore, the adversary is required to keep the revealed portion of the graph connected at all times. We present an algorithm that achieves 2-competitiveness on trees. We also present algorithms that achieve 2.5-competitiveness on cactus graphs, $(t - 1)$ -competitiveness on $K_{1,t}$ -free graphs, and $\Theta(\sqrt{\Delta})$ for maximum degree Δ graphs. We show that all of those competitive ratios are tight. Then, we study several more general classes of graphs, such as threshold, bipartite planar, and series-parallel graphs, and show that they do not admit competitive algorithms (i.e., when competitive ratio is independent of the input size). Previously, the dominating set problem was considered in a different input model (often together with the restriction of the input graph being always connected), where a vertex is revealed alongside its restricted neighborhood: those neighbors that are among already revealed vertices. Thus, conceptually, our results quantify the value of knowing the entire neighborhood at the time a vertex is revealed as compared to the restricted neighborhood. For instance, it was known in the restricted neighborhood model that 3-competitiveness is optimal for trees, whereas knowing the neighbors allows us to improve it to 2-competitiveness.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Dominating set, online algorithms, competitive ratio, trees, cactus graphs, bipartite planar graphs, series-parallel graphs, closed neighborhood

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.57

Related Version *Full Version*: <https://arxiv.org/abs/2105.00299>

Funding This research is supported NSERC.

1 Introduction

Given an undirected simple graph $G = (V, E)$, a subset of vertices $D \subseteq V$ is called *dominating* if every vertex of V is either in D or is adjacent to some vertex in D . In the well-known \mathcal{NP} -hard dominating set problem, the goal is to find a dominating set of minimum cardinality. We study this problem in the online setting, where a graph is revealed one node at a time. When a node is revealed its entire neighborhood is revealed as well. An algorithm is required to make an irrevocable decision on whether to include the newly revealed vertex into the dominating set the algorithm is constructing or not. This decision must be made before the next vertex is revealed. Performance of an online algorithm is measured against an optimal



© Hovhannes A. Harutyunyan, Denis Pankratov, and Jesse Racicot;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 57; pp. 57:1–57:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

offline algorithm, i.e., an algorithm that knows the entire input in advance and has infinite computational resources. This measure is captured by the notion of competitive ratio and analysis, which is made precise below. For now, it suffices to note that competitive ratio is analogous to approximation ratio in the offline setting.

The dominating set problem has important practical and theoretical applications, such as establishing surveillance service ([1]), routing and transmission services in (wireless) networks ([5]), as well as broadcasting ([7, 8]). While the dominating set problem and its variants (connected dominating set, independent dominating set, weighted dominating set, etc.) have been extensively studied in the offline setting [1, 9, 11, 15, 16, 17], this problem has received little attention in the online algorithms community. The current paper attempts to fill in this gap, while making a quantitative comparison with another online model for dominating set.

Online dominating set problem has been studied in the vertex arrival model by Boyar et al. [3]. In that model, when a vertex is revealed only restricted neighborhood of that vertex is revealed as well, namely, those neighbors that appear among previously revealed vertices. Moreover, in the model considered by Boyar et al. decisions are only partially irrevocable, i.e., when a vertex arrives an algorithm may add this vertex together with *any of its neighbors from the restricted neighborhood* to the dominating set. Thus, the decision to include a vertex is irrevocable, while the decision not to include a vertex is only partially irrevocable – an algorithm has a chance to reconsider when any yet unrevealed neighbors arrive. The catch is that the algorithm does not know the input size and has to maintain a dominating set at all times. In the model considered in this paper, all decisions (to include or exclude a vertex from a dominating set) are irrevocable. Boyar et al. [3] considered the online dominating set problem in two settings, namely, with the restriction of an adversary being forced to maintain an always connected graph and without this restriction. For the fairness of comparison, when we talk about Boyar et al. results we refer to their results for the always-connected setting¹. To summarize, on one hand, our model is stronger for the adversary since it forces the algorithm to make an irrevocable decision at each step. On another hand, our model is weaker for the adversary than the model of Boyar et al. in the aspect of the adversary being forced to reveal all neighbors of a newly revealed vertex at once. Thus, our results when compared to those of the vertex arrival model can be viewed as quantifying the value of getting to know all neighbors of a vertex at the time of its revelation.

Perhaps somewhat surprisingly, we discover in several results that the benefit of knowing all neighbors outweighs the drawbacks of fully irrevocable decisions. Our results are summarized below, but in particular we show that in our model Δ -bounded degree graphs admit $O(\sqrt{\Delta})$ online algorithms, while Boyar et al. show that $\Omega(\Delta)$ is necessary in their model. Similarly, we analyze a 2-competitive algorithm for trees, while Kobayashi [13] shows a lower bound of 3 in the vertex arrival model. Our degree upper bound implies that $O(\sqrt{n})$ competitive ratio is tight for general graphs, whereas Boyar et al. showed the lower bound of $\Omega(n)$ in the vertex arrival model. This paints a picture that knowing all the neighbors improves not only precise constants, when graph classes allow for small competitive ratio algorithms, but also give asymptotic improvements for more “challenging” graph classes for algorithms.

Prior to summarizing our results, we give a brief overview of competitive analysis framework. For more details, an interested reader should consult excellent books [2, 14] and references therein. Let ALG be an algorithm for the online dominating set problem.

¹ In our model, two natural definitions of always-connected restriction are possible: (i) with respect to all vertices that the algorithm is aware of at any particular moment (this includes vertices that have arrived and their neighbors that have not yet arrived), and (ii) with respect to only those vertices that have arrived. Our work is in setting (ii). This distinction is absent in the vertex arrival model.

Let $ALG(G, \sigma)$ denote the set of vertices that are selected by ALG on the input graph G with its vertices revealed according to the order σ . We sometimes abuse the notation and omit G or σ (or both) when they are clear from the context. Abusing notation even more, we sometimes write $ALG(G, \sigma)$ to mean $|ALG(G, \sigma)|$. Similar conventions apply to an offline optimal solution denoted by OPT . We say that ALG has *strict competitive ratio* c if $ALG \leq c \cdot OPT$ on all inputs. We say that ALG has *asymptotic competitive ratio* c (or, alternatively, that ALG is c -competitive) if $\limsup_{OPT \rightarrow \infty} \frac{ALG}{OPT} \leq c$. The *competitive ratio* of ALG is the infimum over all c such that ALG is c -competitive. When we simply write “competitive ratio” we typically mean “asymptotic competitive ratio” unless stated otherwise.

We shall consider performance of algorithms with respect to restricted inputs, specified by various graph classes, such as trees, cactus graphs, series-parallel, etc. The above definitions of competitive ratios can be modified by restricting them to inputs coming from certain graph classes. We denote the competitive ratio of an algorithm ALG with respect to the restricted graph class $CLASS$ by $\rho(ALG, CLASS)$.

The following is a summary of our contributions with the section numbers where the results appear. Due to space considerations some of our results have been moved to appendix:

- tight competitive ratio 2 on trees (Section 3.1);
- tight competitive ratio $\frac{5}{2}$ on cactus graphs (Section 3.2);
- tight competitive ratio $\Theta(\sqrt{\Delta})$ on maximum degree Δ graphs (Section 3.3);
- tight competitive ratio $t - 1$ on $K_{1,t}$ -free graphs (Section 3.4);
- tight competitive ratio $\Theta(\sqrt{n})$ for threshold graphs (Section B.1), planar bipartite graphs (Section B.2), and series-parallel graphs (Section B.3).

We note that all our upper bounds are in terms of strict competitive ratios, and all our lower bounds, with the exception of $K_{1,t}$ -free graphs, are in terms of asymptotic competitive ratios.² Most of our upper bounds are established by charging arguments. Our charging schemes are natural and to analyze them we establish several combinatorial properties of relevant graph classes. We suspect that these (or similar) techniques can be used to extend the results to other graph classes, such as almost-tree(k). Our main contribution is conceptual: we begin a systematic study of a well known \mathcal{NP} -hard problem in an online setting that hasn’t been extensively considered before and which allows quantifying how much extra information about the neighborhood helps the competitive ratio.

2 Preliminaries

In this section we describe definitions and establish notations that will be used frequently in the rest of the paper. Let $G = (V, E)$ be a connected undirected graph on $n = |V| \geq 1$ vertices. The *closed neighborhood* of a subset of vertices $S \subseteq V$, denoted by $N[S]$, is defined as $S \cup \{v \in V \mid \exists u \in S, \{u, v\} \in E\}$.

The vertices V are revealed online in order (v_1, \dots, v_n) . Since we consider the online input model where vertices are revealed alongside their neighbors, we distinguish between two notions: those vertices that are revealed by a certain time and those that are visible. More precisely, we have the following:

² With the small caveat that the performance ratio for threshold graphs is measured as a function of input size for reasons provided later.

► **Definition 1.**

- v_i is revealed by time j if $i \leq j$.
- v_j is visible at time i if it is either revealed by time i or it is adjacent to some vertex revealed by time i .
- R_i denotes the set of all vertices revealed by time i .
- V_i denotes the vertices visible at time i (i.e. $V_i = N[R_i]$).

The adversary chooses the graph G as well as the revelation order of vertices; however, the adversary is restricted to those revelation orders that guarantee that the induced subgraph on R_i is connected for all i . Thus, we observe that the process of revelation of a graph by the adversary is a natural generalization of the breadth-first search (BFS) and depth-first search (DFS) explorations of the graph. Thus, we can define the *revelation tree* analogous to BFS and DFS trees. We need the following observation first:

► **Observation 2.** If $v_j \in V_i \setminus V_{i-1}$ with $j > i \geq 2$ then v_i is the unique neighbor of v_j at time i .

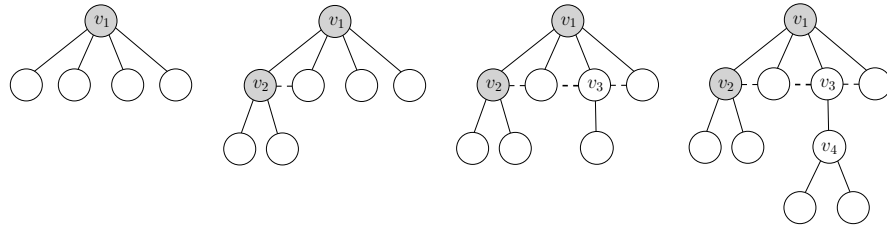
In the preceding observation, we say that v_j is a *child* of v_i and that v_i is the *parent* of v_j . The edge $\{v_i, v_j\}$ is called a *tree edge*. The subgraph induced on the tree edges is the *revelation tree*. Any edge $\{u, v\}$ where u is not the parent of v nor v the parent of u is called a *cross edge*.

After the vertex v_i is revealed together with its closed neighborhood $N[v_i]$, an online algorithm ALG must make a decision $d_i \in \{0, 1\}$, which indicates whether the algorithm takes this vertex to be in the dominating set or not. For a given online algorithm ALG we define the following:

► **Definition 3.**

- $S_i = \{v_k \mid d_k = 1, 1 \leq k \leq i\}$ is the set of revealed vertices selected by ALG after i decisions where $S_0 = \emptyset$.
- $D_i = N[S_i]$ is the set of visible vertices that are dominated after i decisions.
- $U_i = V_i \setminus D_{i-1}$ is the set of visible vertices undominated immediately before decision d_i where $U_0 = \emptyset$.

A series of figures are provided below which illustrate the preceding definitions. For these figures, and all others in this paper, the convention is that vertices that are shaded in gray are those selected by ALG , vertices with thicker boundaries belong to OPT , an edge that is dashed is a cross edge, and all the solid edges are tree edges.



■ **Figure 1** An example of vertices v_1, v_2, v_3, v_4 from some input graph being revealed in that order (from left to right). Empty vertices in this figure are visible but not yet revealed. The adversary must maintain the connectivity of revealed vertices (ignoring visible but not yet revealed vertices) at all times. The process continues until all vertices are revealed. An edge that is dashed is a cross edge and one that is solid is a tree edge.

Since an algorithm makes irrevocable decisions and must produce a feasible solution, there may be situations where an algorithm is forced to select a vertex v_j to be in the dominating set. This happens because v_j is the “last chance” to dominate some other vertex v_i . In this case, we say that v_j saves v_i or that v_j is the savior of v_i . Note that it is possible for a vertex v_j to save itself. The following definition makes the notion of “saving” precise.

► **Definition 4.** A vertex v_j saves a vertex v_i if $j = \max\{k \mid v_k \in N[v_i]\}$ and $N[v_i] \setminus \{v_j\}$ contains no vertices from S_{j-1} . Let $s(v_j)$ denote the set of vertices that v_j saves.

Observe that if a vertex is saved then it must be that every one of its neighbors (itself included) had a chance to dominate the said vertex.

► **Observation 5.** If v_i is saved then $v_i \in N[v_j] \cap U_j$ for any $v_j \in N[v_i]$.

All our upper bounds are established by either a GREEDY algorithm or a k -DOMINATE algorithm for some fixed integer value of parameter k :

- The algorithm GREEDY selects a newly revealed vertex if and only if the vertex is not currently dominated. Using the notation introduced above, GREEDY selects $v_i, i \geq 1$ if and only if $v_i \in U_i$.
- The algorithm k -DOMINATE (for some fixed integer parameter k) selects a newly revealed vertex if and only if either (1) the vertex has at least k undominated neighbors, or (2) the vertex saves at least one other vertex. Using the notation introduced before, v_i is selected if and only if either (1) $|N(v_i) \cap U_i| \geq k$, or (2) $|s(v_i)| \geq 1$.

Both GREEDY and k -DOMINATE give rise to rather efficient offline algorithms so that any of the positive results given in this paper may be realized as efficient offline approximation algorithms.

3 Competitive Graph Classes

3.1 Trees

In this section we establish the tight bound of 2 on the best competitive ratio when the input graph is restricted to be a tree. The upper bound is achieved by the 2-DOMINATE algorithm and is proved in Theorem 7 below. The lower bound on all online algorithms is established in Theorem 6. Within this section all of the formal statements implicitly assume that the input is a tree. We begin the section by proving the lower bound.

► **Theorem 6.** $\rho(\text{ALG}, \text{TREE}) \geq 2$ for any algorithm ALG .

Proof. Consider an arbitrary small $\epsilon > 0$. We will give an adversarial input that guarantees that $\text{ALG} \geq (2 - \epsilon)\text{OPT}$. Let $k = \lceil \frac{3}{\epsilon} \rceil \geq 4$. At the start, the adversary reveals v_1 with k children $\{c_1, \dots, c_k\}$. Then we start the process described in the next paragraph at c_1 . The process can terminate in two ways: (i) ALG stops selecting vertices to be in the dominating set, or (ii) ALG selects k vertices revealed after c_1 (inclusive). If the process terminates because of (i), then the adversary restarts the process at child c_2 of v_1 . The process again terminates either with (i) or (ii) with respect to c_2 . If it is due to (i), then the adversary restarts the process at c_3 , and so on. If the process terminates with (ii) with respect to c_i then we reveal c_j for $j > i$ as leaves of v_1 .

Next, we describe the process with respect to c_i . The adversary reveals c_i with 2 children and if ALG selects c_i then exactly one child of c_i is revealed with two additional children. If ALG selects the child then one of its children is revealed with two additional children, and so on. Let j_i be the number of these vertices that are selected by ALG . This process

terminates only if ALG stops selecting these vertices with two children ($j_i < k$) or when ALG selects k of them ($j_i = k$). At this point the subtree grown at c_i has some revealed vertices as well as visible, but not yet revealed vertices. To finish revealing the entire subtree, the adversary proceeds as follows.

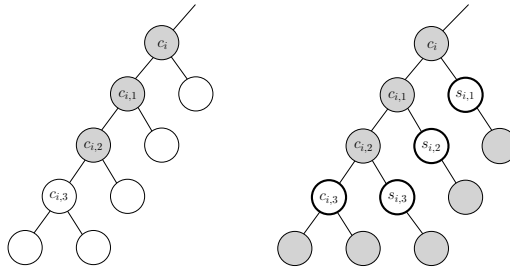
If $j_i < k$ then the two children on the $(j_i + 1)$ 'st vertex are revealed to be leaves. Moreover, each of the j_i selected vertices have exactly one visible child that is not yet revealed. Reveal those j_i children, called *support vertices*, with an additional leaf child (i.e. the child is revealed to be a leaf after its parent is revealed). Including the 2 children of the $(j_i + 1)$ 'st vertex ALG must select at least $j_i + 2$ additional vertices to dominate these leaves for a total of $j_i + (j_i + 2) = 2(j_i + 1)$ selected vertices in this subtree. In this case, OPT can select the support vertices together with the $(j_i + 1)$ 'st vertex for a total $j_i + 1$ vertices to dominate the entire subtree.

If $j_i = k$ the procedure to finish revealing the entire subtree at c_i is similar: the k 'th vertex children are both revealed to be leaves and each of the other $k - 1$ selected vertices has the other child become a support vertex, i.e., revealed with an additional leaf child. The performance is similar here but ALG is not forced to select the two children of the k 'th vertex so ALG selects at least $k + (k - 1) = 2k - 1$. In this case, OPT needs only select the k 'th vertex together with the support vertices for a total of k vertices to dominate the subtree.

To finish the analysis, we consider the following two cases:

Case 1: for all i we have $j_i < k$. Then $ALG \geq 2(j_i + 1)$ on each subtree whereas $OPT \leq j_i + 1$ on each subtree. Summing over all subtrees and remarking that OPT might select v_1 we obtain that $ALG/OPT \geq (\sum 2(j_i + 1)) / (1 + \sum(j_i + 1)) \geq 2 - 2/k \geq 2 - \epsilon$.

Case 2: there exists ℓ such that $j_\ell = k$. Then OPT selects $j_i + 1$ vertices for $i < \ell$, k vertices for $i = \ell$, 0 vertices for $i > \ell$ per subtree, plus v_1 . Whereas ALG selects at least $2(j_i + 1)$ for $i < \ell$, $2k - 1$ for $i = \ell$, and 0 for $i > \ell$. By a similar calculation to **Case 1**, we obtain that $ALG/OPT \geq 2 - 3/k \geq 2 - \epsilon$. ◀



■ **Figure 2** An example of the process described in Theorem 6 where ALG selects $j_i = 3$ vertices on the subtree rooted at c_i . The top depicts the subtree immediately after revealing $c_{i,3}$ whereas the bottom shows the entirely revealed subtree.

Now that we have established an asymptotic lower bound of 2 for any algorithm we show that 2-DOMINATE is 2-competitive.

► **Theorem 7.** $\rho(2\text{-DOMINATE}, TREE) = 2$.

High level overview of the proof. Consider an arbitrary input $T = (V, E)$ on $n \geq 3$ vertices and let OPT denote a minimum dominating set of T which contains no vertices of degree 1 (i.e. any such vertex can be exchanged for its only neighbor). Recall that S is the set of vertices selected by 2-DOMINATE. Initially, we assign charge 1 to each vertex v in S and

charge 0 to each vertex v not in S . Thus, $|S| = \sum_{v \in S} ch_{init}(v)$ where $ch_{init}(v)$ denotes the initial charge of v . With a charging scheme described shortly, we spread the charge from the vertices in S to the vertices of V . Let $ch(v)$ denote the new charge associated with vertex v . We extend the functions ch_{init} and ch to subsets of vertices linearly, e.g., for $W \subseteq V$ we have $ch(W) = \sum_{v \in W} ch(v)$. We shall demonstrate that the procedure of spreading the charge satisfies two properties:

1. conservation property: $\sum_v ch_{init}(v) = \sum_v ch(v)$ meaning that the total charge is preserved; and
2. *OPT*-concentration property: for each $v \in OPT$ we have $ch(N[v]) \leq 2$.

With these two properties it follows that $2\text{-DOMINATE} \leq \sum_v ch_{init}(v) = \sum_v ch(v) \leq \sum_{v \in OPT} ch(N[v]) \leq 2OPT$, so 2-DOMINATE is strictly 2-competitive.

Before we proceed with this plan, we make a couple of useful observations:

► **Lemma 8.** *There are no cross edges incident on any vertex v_i . In particular, any vertex v_i has at most one neighbor before it is revealed.*

► **Corollary 9.** *If $\deg(v_i) \geq 3$ then $v_i \in S$.*

Now, we are ready to present formal details of the above plan. We spread the charges according to the following rule:

Consider any $v_i \in S$ with $X_i = N[v_i] \cap U_i$. Remarking that $X_i \neq \emptyset$ we then give each vertex in X_i an equal charge of $\frac{1}{|X_i|}$. That is, a vertex selected by 2-DOMINATE spreads its charge evenly to all the newly dominated vertices in its closed neighborhood. We say that each vertex in X_i is charged by v_i .

► **Observation 10.** *Every vertex is charged by exactly one vertex.*

The preceding observation immediately implies that any vertex has charge at most 1. This observation is tight in the sense that, on certain inputs, there are vertices with charge equal to 1. A vertex with charge 1 is a rather special case though. Suppose that a vertex v_i receives charge 1 from a vertex v_j where v_i and v_j are not necessarily distinct. Therefore we have that $|X_j| = |N[v_j] \cap U_j| = 1$ which implies that $|N(v_j) \cap U_j| \leq |N[v_j] \cap U_j| = 1 < 2$. Therefore when v_i receives charge it must be from a vertex $v_j \in S$ that was selected due to the “saviour” rule of 2-DOMINATE . Hence, v_j must have saved a vertex, and only one vertex since $|X_j| = 1$. Ultimately we conclude that if v_i has charge 1 then it must be saved by some vertex v_j where $X_j = \{v_i\}$ (this does not exclude the possibility that $v_i = v_j$). If v_i does not meet this condition then it must have charge at most $\frac{1}{2}$.

► **Lemma 11.** *If v_i and v_j both have charge equal to 1 then they share no common neighbors.*

Proof. Suppose for the sake of deriving a contradiction that $v_{i'}$ were a common neighbor of v_i and v_j . Since v_i is saved, by Observation 5 it must be that $v_i \in N(v_{i'}) \cap U_{i'}$. Similarly, we have that $v_j \in N(v_{i'}) \cap U_{i'}$. That is, $|N(v_{i'}) \cap U_{i'}| \geq 2$ and thus $v_{i'} \in S$. Moreover, $X_{i'} = N[v_{i'}] \cap U_{i'}$ contains v_i and v_j . In particular, we have that $|X_{i'}| \geq 2$ with $v_i, v_j \in X_{i'}$ and therefore v_i and v_j receive charge no larger than $\frac{1}{2}$, a contradiction. ◀

► **Lemma 12.** *If v_i and v_j both have charge equal to 1 then they are not adjacent.*

Proof. It is easy to see that v_1 cannot have charge 1 on any input with at least 2 vertices. Therefore we safely assume that $1 < i < j$ such that both v_i and v_j have a parent. We assume for the sake of deriving a contradiction that v_i and v_j are adjacent.

Now, since both v_i and v_j have charge 1 it follows that they are both saved vertices. First we show that both $v_i, v_j \notin S$. Notice that any saved vertex v_k has the property that $|N[v_k] \cap S| = 1$. Therefore, if we assume by way of contradiction that $v_i \in S$ we obtain that $N[v_i] \cap S = N[v_j] \cap S = \{v_i\}$ and therefore v_i saves itself and v_j . This yields that $X_i = N[v_i] \cap U_i$ contains v_i and v_j . In particular, we have that $|X_i| \geq 2$ with $v_i, v_j \in X_i$ and therefore v_i and v_j receive charge no larger than $\frac{1}{2}$, a contradiction. An identical argument will yield that $v_j \notin S$.

Therefore it must be that v_i is saved by some vertex $v_{i'}$ with $i' \notin \{i, j\}$. Moreover, we must have $i < j < i'$ since $i < j$ by assumption and $i' = \max\{k \mid v_k \in N[v_i]\}$. This implies that both $v_j, v_{i'}$ are children of v_i by Observation 8 yielding that $|N(v_i) \cap U_i| \geq 2$ but v_i cannot be in S . ◀

From the two preceding lemmas we have the immediate corollary.

► **Corollary 13.** *For any vertex v_i , at most one vertex in $N[v_i]$ has charge 1.*

Now, we finish the proof of 2-competitiveness of 2-DOMINATE on trees.

Proof of Theorem 7. The lower bound follows from Theorem 6. Let $v_i \in OPT$ be an arbitrary vertex in OPT . We consider two cases (1) $\deg(v_i) = 2$ or (2) $\deg(v_i) \geq 3$.

Case 1: Suppose that $\deg(v_i) = 2$ and hence $|N[v_i]| = 3$. By Corollary 13 it follows that at most one vertex in $N[v_i]$ has charge 1. If no vertices in $N[v_i]$ have charge 1 then $ch(x) \leq \frac{1}{2}$ for each $x \in N[v_i]$ and we obtain that $\sum_{x \in N[v_i]} ch(x) \leq 3(\frac{1}{2}) < 2$. If there is exactly one vertex $x' \in N[v_i]$ with charge 1 we therefore obtain that $\sum_{x \in N[v_i]} ch(x) =$

$$\sum_{x \in N[v_i] \setminus \{x'\}} ch(x) + ch(x') \leq \frac{2}{2} + 1 = 2.$$

Case 2: Suppose that $\deg(v_i) \geq 3$. By Corollary 9 it follows that $v_i \in S$ with at least 2 children. Let $C_i = V_i \setminus V_{i-1}$ denote the children of v_i and remark that $C_i \subseteq X_i$. That is, each child of v_i is charged by v_i and only v_i . Therefore the children of v_i can receive at most the full initial charge on v_i and thus attribute a charge of at most 1.

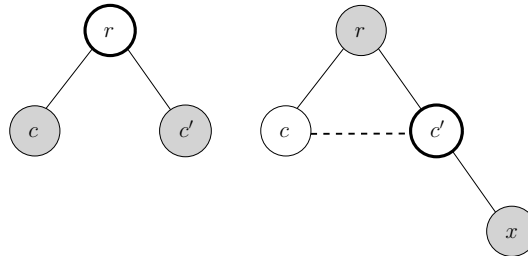
Now we claim that any vertex in $N[v_i] \setminus C_i$ has a charge of at most $\frac{1}{2}$. Indeed, suppose a vertex $v_{i'} \in N[v_i] \setminus C_i$ has charge 1 then it must be saved by v_i since $|N[v_{i'}] \cap S| = 1$ for any saved vertex $v_{i'}$. That is, there is exactly one vertex in its closed neighborhood that is selected and since v_i is selected it must be v_i . Thus, we must have that $v_{i'} \in X_i$ but since $C_i \subseteq X_i$ we know that $|X_i| \geq 2$ and thus $v_{i'}$ receives a charge of no more than $\frac{1}{2} < 1$, contradicting our assumption that $v_{i'}$ has charge 1.

Thus, by remarking that $|N[v_i] \setminus C_i| \leq 2$ we obtain that $\sum_{x \in N[v_i]} ch(x) = \sum_{v_j \in C_i} ch(v_j) + \sum_{v_{i'} \in N[v_i] \setminus C_i} ch(v_{i'}) \leq 1 + 2(\frac{1}{2}) = 2$ as desired. ◀

3.2 Cactus Graphs

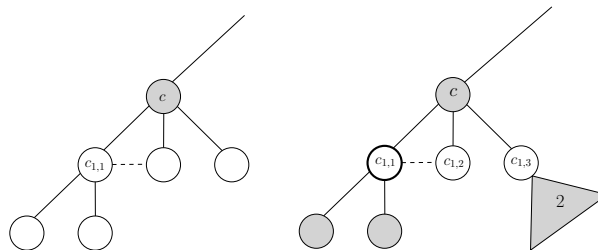
A graph G is said to be a cactus graph if it is connected and every edge lies on at most one cycle. Hedetniemi, Laskar, and Pfaff [10] provide an exact offline algorithm that runs in linear time for finding a minimum dominating set of a cactus graph. Of course, an efficient offline algorithm does not guarantee that an online algorithm can perform well but fortunately, cactus graphs are a class of graphs for which an online algorithm can achieve

constant competitive ratio. In this section, we show that 2-DOMINATE is $\frac{5}{2}$ -competitive when inputs are restricted to cactus graphs, and that this is as well as any algorithm can perform. Within this section all of the formal statements implicitly assume that the input is a cactus graph.



■ **Figure 3** The cactus 2-gadget : The leftmost figure depicts the case where *ALG* does not select the root *r* and rightmost depicts the case where *ALG* selects *r*.

Before presenting a lower bound of $\frac{5}{2}$ on all online algorithms we describe a gadget that is used in the proof. The gadget itself is a cactus graph on $3 \leq n \leq 4$ vertices with the property that *OPT* selects exactly 1 vertex and any algorithm *ALG* selects at least 2 vertices. Consider revealing a root vertex *r* with 2 children *c* and *c'*. If *ALG* does not select *r* then both *c, c'* are revealed as only adjacent to *r* and *ALG* must select both whereas *OPT* selects only *r*. If *ALG* does select *r* then *c* is revealed as adjacent to *c'*, and *c'* is revealed with an additional child *x*. The vertex *x* is adjacent only to *c'* and thus *ALG* must select at least one of *c', x* whereas *OPT* selects only *c'* (both cases are depicted in Figure 3). Given any input cactus graph with a visible vertex *r* not yet revealed this gadget can be constructed with *r* as the root. Within the proof of the lower bound we call this a 2-gadget.



■ **Figure 4** The case described in Theorem 14 where *ALG* does not select $c_{1,1}$.

► **Theorem 14.** $\rho(\text{ALG}, \text{CACTUS}) \geq \frac{5}{2}$ for any algorithm *ALG*.

Proof. Consider an arbitrary small $\epsilon > 0$ and let $k = \lceil \frac{4}{\epsilon} \rceil \geq 5$. We will give an adversarial input that guarantees that $\text{OPT} \geq k$ and $\text{ALG} \geq (\frac{5}{2} - \epsilon)\text{OPT}$. To begin the input, the adversary reveals v_1 with k children $\{c_1, \dots, c_k\}$. Then we run an adversarial process starting with the child c_1 of v_1 . The process consists of rounds, where each round increases *OPT* by 2 while increasing *ALG* by 5. The process might terminate for one of two reasons: either (i) we guarantee strict competitive ratio at least $5/2$ on the subcactus rooted at c_1 , or (ii) k rounds starting at c_1 elapse. If the process terminates because of (i), then the adversary restarts the process at child c_2 of v_1 . The process again terminates either with (i) or (ii) with respect to c_2 . If it is due to (i), then the adversary restarts the process at c_3 , and so on. If the process terminates with (ii) with respect to c_i then we reveal c_j for $j > i$ as leaves. Below we describe the process starting at a child of v_1 although the first round of the process differs from the others that follow.

We now describe the first round starting at a child c of v_1 . Initially, we reveal c with 3 children. If ALG does not select c then each child of c is revealed as leaf and ALG must select all 3 children whereas OPT selects c . Suppose then that ALG selects c and let $c_{1,1}, c_{1,2}, c_{1,3}$ be the three children of c . Reveal $c_{1,1}$ as adjacent to $c_{1,2}$ along with 2 additional children. If ALG does not select $c_{1,1}$ then the children of $c_{1,1}$ are revealed as leaves, forcing ALG to select them and $c_{1,3}$ is revealed as the root of a 2-gadget ($c_{1,2}$ is revealed with no additional neighbors). Thus, $\frac{ALG}{OPT} \geq \frac{5}{2}$ in this case (see Figure 4). If instead ALG selects $c_{1,1}$ then $c_{1,2}$ and $c_{1,3}$ are revealed as the roots of two distinct 2-gadgets and since c is dominated by v_1 (we assume that $v_1 \in OPT$) we have that $\frac{ALG}{OPT} \geq \frac{5}{2}$ on this subcactus (excluding $c_{1,1}$) thus far (see Figure 5a). At this point, $c_{1,1}$ is selected by ALG and we start the second round (which is described below) with $c_{1,1}$ as the root. Every round that follows will be the same as the second and requires a root selected by ALG which has two children.

The second round starts at a selected root $c_{1,1}$ and we let $c_{2,1}, c_{2,2}$ be the 2 children of $c_{1,1}$. We reveal $c_{2,1}$ as adjacent to $c_{2,2}$ with 2 children $c_{3,1}, c_{3,2}$. If ALG does not select $c_{2,1}$ then $c_{3,1}, c_{3,2}$ are revealed as leaves and ALG selects $c_{1,1}, c_{3,1}, c_{3,2}$ and OPT can select $c_{2,1}$ for a performance of 3 along with the running performance of $\frac{5}{2}$ (see Figure 5b). If ALG does select $c_{2,1}$ then $c_{3,1}$ is revealed as adjacent to $c_{3,2}$ with two children $c_{4,1}, c_{4,2}$. If ALG does not select $c_{3,1}$ then $c_{4,1}, c_{4,2}$ are revealed as leaves and $c_{2,2}$ is revealed with an additional leaf neighbor $l_{2,2}$ so that ALG must select at least one of $c_{2,2}, l_{2,2}$. Thus, ALG here selects $c_{1,1}, c_{2,1}, c_{4,1}, c_{4,2}$ and at least one of $c_{2,2}, l_{2,2}$ whereas OPT can select $c_{3,1}$ and $c_{2,2}$ for a performance of $\frac{5}{2}$ (see Figure 6a). If instead ALG selects $c_{3,1}$ (thus far $c_{1,1}, c_{2,1}$ and $c_{3,1}$ are all selected) then $c_{2,2}$ is revealed with an additional leaf neighbor $l_{2,2}$ so that ALG must select at least one of $c_{2,2}, l_{2,2}$, and $c_{3,2}$ is revealed as the root of a 2-gadget so that $\frac{ALG}{OPT} \geq \frac{5}{2}$ on the subcactus thus far (excluding $c_{3,1}$) and we repeat the trap with $c_{3,1}$ as the selected root (see Figure 6b).

Let $j_i \geq 1$ denote the number of rounds that passed in the adversarial process starting at the child c_i . To finish the analysis, we consider the following two cases:

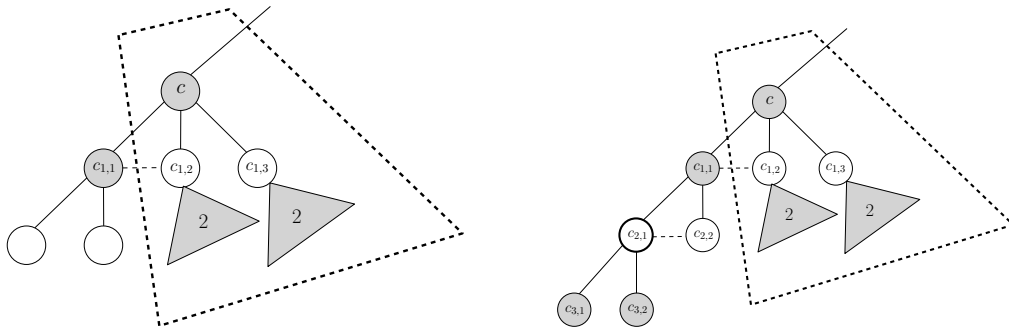
Case 1: For all i we have that $j_i < k$. Then $ALG \geq 5j_i$ on each subcactus whereas $OPT \leq 2j_i$ on each subcactus³. Summing over all subcacti and remarking that OPT selects v_1 we obtain that $ALG/OPT \geq (\sum 5j_i) / (1 + \sum 2j_i) \geq \frac{5}{2} - \frac{5}{2k} \geq \frac{5}{2} - \epsilon$.

Case 2: There exists ℓ such that $j_\ell = k$. In this case, there is an additional vertex $c_{j,1}$ with $j = 3(k-1)$ that was selected by ALG and must also be selected by OPT . (i.e. c_j is the root where a $(k+1)$ 'st round could start). Therefore, OPT selects $2j_i$ vertices for each process on child c_i with $i < \ell$, $2k+1$ vertices for $i = \ell$, 0 vertices for $i > \ell$ plus v_1 . Whereas ALG selects at least $5j_i$ for $i < \ell$, $5k+1$ for $i = \ell$, and 0 for $i > \ell$. Ultimately, we obtain that $ALG/OPT \geq (\sum 5j_i + 5k + 1) / (\sum 2j_i + 2k + 2) \geq \frac{5}{2} - 4/k \geq \frac{5}{2} - \epsilon$. ◀

► **Theorem 15.** $\rho(2\text{-DOMINATE}, \text{CACTUS}) = \frac{5}{2}$.

The proof can be viewed as an adaptation of our proof for trees to cactus graphs. We use a charging argument similar to the one given in the section on trees. Initially, a charge of 1 is given for each $v \in S$, the charge on each vertex is then spread to certain neighbors, and we then show that $\sum_{x \in N[v_i]} ch(x) \leq \frac{5}{2}$ for each $v_i \in OPT$. We spread the charge according

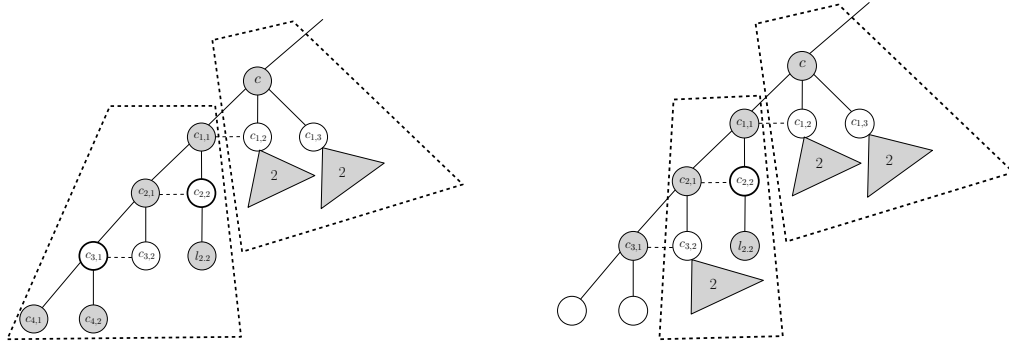
³ We have omitted the cases where ALG does not select the root c_i . These cases result in ALG selecting 3 vertices on the subcacti with OPT selecting only 1 and the result clearly still holds in this case.



(a) *ALG* does select $c_{1,1}$. The enclosed region contributes a performance of $\frac{5}{2}$. A trap is continued in this case with the root $c_{1,1}$.

(b) *ALG* does not select $c_{2,1}$.

■ **Figure 5** Two cases described in Theorem 14.



(a) *ALG* does not select $c_{3,1}$. The enclosed regions each contribute a performance of $\frac{5}{2}$.

(b) *ALG* does select $c_{3,1}$. The enclosed regions each contribute a performance of $\frac{5}{2}$. The trap that was used on a selected root $c_{1,1}$ is repeated with $c_{3,1}$ as the selected root.

■ **Figure 6** Two more cases described in Theorem 14.

to the same rule given in the preceding section and recall that Observation 10 (each vertex receives a new charge from one other vertex) still holds. In the analysis of how the charge gets reallocated, the structure of the underlying graph is of paramount importance. We begin with an analogue to Lemma 8.

► **Lemma 16.** *There is at most one cross edge incident on any v_i . In particular, v_i has at most 2 neighbors before it is revealed.*

Proof. Suppose that $v_i \neq v_1$ since the statement is clearly true for $v_i = v_1$. Suppose for the sake of deriving a contradiction that, at time $i - 1$, v_i has three neighbors v_h, v_{i_1}, v_{i_2} where v_h is the parent of v_i and $\{v_i, v_{i_1}\}, \{v_i, v_{i_2}\}$ are cross edges. Notice that v_{i_1} is visible at time $i - 1$ as otherwise would imply that $\{v_i, v_{i_1}\}$ were a tree edge. Thus, at time $i - 1$, v_{i_1} is visible and there is only one tree edge incident on v_i . In particular, this implies that there is a path consisting entirely of tree edges from v_{i_1} to v_h where said path does not contain the edge $\{v_h, v_i\}$ since it does not pass through v_i nor does it contain the edges $\{v_i, v_{i_1}\}, \{v_i, v_{i_2}\}$ since they are cross edges. Thus, by adding edges $\{v_h, v_i\}, \{v_i, v_{i_1}\}$ to this path we obtain a cycle (in the completely revealed input graph) that contains the edge $\{v_h, v_i\}$ but does not

contain the edge $\{v_i, v_{i_2}\}$. A similar argument yields that there is a path consisting of tree edges from v_{i_2} to v_h that does not contain the edges $\{v_h, v_i\}, \{v_i, v_{i_1}\}, \{v_i, v_{i_2}\}$ and hence by adding edges $\{v_h, v_i\}, \{v_i, v_{i_2}\}$ we obtain a cycle which contains the edge $\{v_h, v_i\}$ but does not contain the edge $\{v_i, v_{i_1}\}$. That is, two distinct cycles that share the common edge $\{v_h, v_i\}$, a contradiction. \blacktriangleleft

Since v_i has at most 2 neighbors before it is revealed then it has at least $\deg(v_i) - 2$ children. The following is analogous to Corollary 9 for trees.

► **Corollary 17.** *If $\deg(v_i) \geq 4$ then $v_i \in S$.*

► **Lemma 18.**

1. *If v_i and v_j both have charge equal to 1 then they share no common neighbors.*
2. *If v_i and v_j both have charge equal to 1 then they are not adjacent.*
3. *For any vertex v_i , at most one vertex in $N[v_i]$ has charge 1.*

Proof.

1. Follows identically to the proof of Lemma 11.
2. First, note that v_1 cannot have charge 1 on any input with at least 2 vertices. Therefore we safely assume that $1 < i < j$ such that both v_i and v_j have a parent. We assume for the sake of deriving a contradiction that v_i and v_j are adjacent.

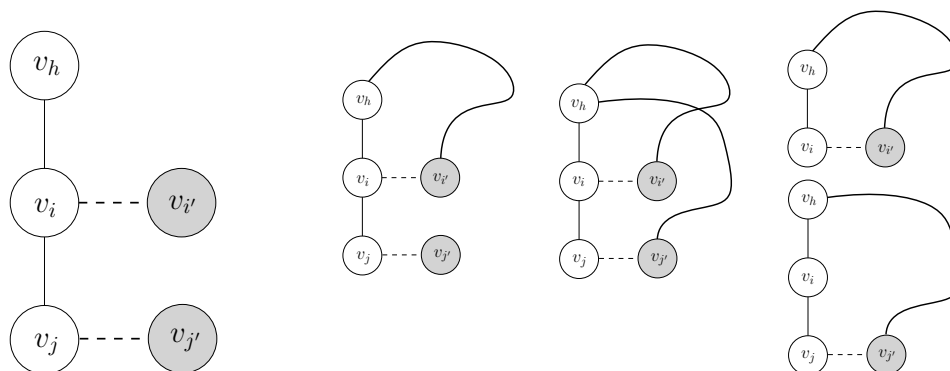
Now, since both v_i and v_j have charge 1 it follows that they are both saved vertices. We first argue that both $v_i, v_j \notin S$. Notice that any saved vertex v_k has the property that $|N[v_k] \cap S| = 1$. Therefore, if we assume by way of contradiction that $v_i \in S$ we obtain that $N[v_i] \cap S = N[v_j] \cap S = \{v_i\}$ and therefore v_i saves itself and v_j . This yields that $X_i = N[v_i] \cap U_i$ contains v_i and v_j . In particular, we have that $|X_i| \geq 2$ with $v_i, v_j \in X_i$ and therefore v_i and v_j receive charge no larger than $\frac{1}{2}$, a contradiction. An identical argument will yield that $v_j \notin S$.

Thus, we assume that v_i is saved by a neighbor $v_{i'}$ and v_j is saved by a neighbor $v_{j'}$ where $i', j' \notin \{i, j\}$. Moreover, $i' \neq j'$ since v_i and v_j can share no common neighbors by part 1. Thus, we have that i, j, i', j' are all distinct with $i < j < i'$ and $i < j < j'$ since $i' = \max\{k \mid v_k \in N[v_i]\}$ and $j' = \max\{k \mid v_k \in N[v_j]\}$. As mentioned above v_i must have a parent v_h where $h < i < j < i'$. Therefore, $\deg(v_i) \geq 3$ and since $v_i \notin S$ it follows by Corollary 17 that $\deg(v_i) = 3$.

We are now in the situation where $v_i, v_j \notin S$ and v_i is incident on exactly 3 edges $\{v_h, v_i\}, \{v_i, v_j\}, \{v_i, v_{i'}\}$ where exactly one of the edges $\{v_i, v_j\}, \{v_i, v_{i'}\}$ is a tree edge (and the other a cross edge). We finish the proof by examining the two cases where **(1)** : $\{v_i, v_{i'}\}$ is a tree edge or **(2)** : $\{v_i, v_j\}$ is a tree edge.

Case 1: Suppose $\{v_i, v_{i'}\}$ is a tree edge so that $v_{i'}$ is a child of v_i . Therefore, $v_{i'} \in C_i \subseteq N(v_i) \cap U_i$, that is, $v_{i'}$ is an undominated neighbor of v_i when v_i is revealed. Since v_j is saved then by Observation 5 it follows that $v_j \in N(v_i) \cap U_i$, that is, v_j is also an undominated neighbor of v_i when v_i is revealed. That is, both $v_{i'}, v_j \in N(v_i) \cap U_i$ implying that $|N(v_i) \cap U_i| \geq 2$ but $v_i \notin S$, a contradiction.

Case 2: Suppose $\{v_i, v_j\}$ is a tree edge so that v_j is a child of v_i . First notice that $\{v_i, v_j\}$ is the only tree edge incident on v_j . Indeed, if there were a tree edge $\{v_j, v_l\}$ then v_l would be the child of v_j . Since v_i is saved we have $v_i \in N(v_j) \cap U_j$ by Observation 5 implying that $|N(v_j) \cap U_j| \geq 2$ but $v_j \notin S$. Thus, we are in the situation depicted in Figure 7a where $\{v_i, v_j\}$ is the only tree edge incident on v_j and by assumption $\{v_h, v_i\}, \{v_i, v_j\}$ are the only two tree edges incident on v_i . Therefore we have a path from $v_{i'}$ to v_h consisting of tree edges where said path does not contain the edges



(a) Case 2 of the second part of Lemma 18. (b) Resolution of the preceding case in Figure 7a. Two cycles sharing the common edge $\{v_h, v_i\}$.

■ **Figure 7** Figures used in the proof of Lemma 18.

$\{v_h, v_i\}, \{v_i, v_{i'}\}, \{v_i, v_j\}, \{v_j, v_{j'}\}$. Thus, by adding edges $\{v_h, v_i\}, \{v_i, v_{i'}\}$ to this path we obtain a cycle (in the completely revealed input) that contains the edge $\{v_h, v_i\}$ but does not contain the edge $\{v_j, v_{j'}\}$. Similarly, there is a path from $v_{j'}$ to v_h consisting of tree edges where said path does not contain the edges $\{v_h, v_i\}, \{v_i, v_{i'}\}, \{v_i, v_j\}, \{v_j, v_{j'}\}$ and by adding edges $\{v_h, v_i\}, \{v_i, v_j\}, \{v_j, v_{j'}\}$ we obtain a cycle (in the completely revealed input) that contains the edge $\{v_h, v_i\}$ but does not contain the edge $\{v_i, v_{i'}\}$. That is, two distinct cycles that share the common edge $\{v_h, v_i\}$, a contradiction.

3. Follows immediately from the previous parts. ◀

Now, we are ready to prove the upper bound for Theorem 15.

Proof of Theorem 15. The lower bound follows from Theorem 14. Let $v_i \in OPT$ be an arbitrary vertex in OPT . We consider two cases (1) $deg(v_i) \leq 3$ or (2) $deg(v_i) \geq 4$.

Case 1: Suppose that $deg(v_i) \leq 3$ and hence $|N[v_i]| \leq 4$. By Lemma 18 part 3 it follows that at most one vertex in $N[v_i]$ has charge 1. If no vertices in $N[v_i]$ have charge 1 then $ch(x) \leq \frac{1}{2}$ for each $x \in N[v_i]$ and we obtain that $\sum_{x \in N[v_i]} ch(x) \leq 4(\frac{1}{2}) = 2 < \frac{5}{2}$. If there is exactly one vertex $x' \in N[v_i]$ with charge 1 we therefore obtain that $\sum_{x \in N[v_i]} ch(x) =$

$$\sum_{x \in N[v_i] \setminus \{x'\}} ch(x) + ch(x') \leq \frac{3}{2} + 1 = \frac{5}{2}.$$

Case 2 : Suppose that $deg(v_i) \geq 4$. By Corollary 17 it follows that $v_i \in S$ with at least 2 children. Let $C_i = V_i \setminus V_{i-1}$ denote the children of v_i and remark that $C_i \subseteq X_i$. That is, each child of v_i is charged by v_i and only v_i . Therefore the children of v_i can receive at most the full initial charge on v_i and thus attribute a charge of at most 1.

Now we claim that any vertex in $N[v_i] \setminus C_i$ has a charge of at most $\frac{1}{2}$. Indeed, suppose a vertex $v_{i'} \in N[v_i] \setminus C_i$ has charge 1 then it must be saved by v_i since $|N[v_{i'}] \cap S| = 1$ for any saved vertex $v_{i'}$. That is, there is exactly one vertex in its closed neighborhood that is selected and since v_i is selected it must be v_i . Thus, we must have that $v_{i'} \in X_i$ but since $C_i \subseteq X_i$ we know that $|X_i| \geq 2$ and thus $v_{i'}$ receives a charge of no more than $\frac{1}{2} < 1$, contradicting our assumption that $v_{i'}$ has charge 1. Thus, by remarking that $|N[v_i] \setminus C_i| \leq 3$ we obtain that $\sum_{x \in N[v_i]} ch(x) = \sum_{v_j \in C_i} ch(v_j) + \sum_{v_{i'} \in N[v_i] \setminus C_i} ch(v_{i'}) \leq 1 + 3(\frac{1}{2}) = \frac{5}{2}$ as desired. ◀

3.3 Graphs of Bounded Degree

We study the problem when the inputs are restricted to graphs of bounded degree. That is, a positive integer $\Delta \geq 2$ is provided to the algorithm beforehand and the adversary is restricted to presenting graphs where every vertex has degree no larger than Δ . The problem of bounded degree graphs was explored in [3] although within the vertex arrival model described earlier. The authors show that a greedy strategy obtains a competitive ratio no larger than Δ and, when inputs are further restricted to be “always-connected” (i.e. each prefix of the input is connected) they provide a lower bound of $\Delta - 2$ for any algorithm.

By definition, any input belonging to our setting is “always-connected” yet the lower bound of $\Delta - 2$ does not apply. In particular, we show that $\lceil \sqrt{\Delta} \rceil$ -DOMINATE is $3\sqrt{\Delta}$ -competitive along with a lower bound of $\Omega(\sqrt{\Delta})$ for any online algorithm, essentially closing the problem in our setting. As previously mentioned, the authors in [12] consider a setting similar to ours where their adversary is not required to reveal visible vertices and they assume that an algorithm has additional knowledge of input size n . In this setting they provide an algorithm that achieves competitive ratio of $\Theta(\sqrt{n})$ for arbitrary graphs. For the upper bound below we follow a proof nearly identical to theirs modulo some minor details and definitions.

► **Definition 19.** A vertex $v_i \in S$ is said to be heavy if $|N(v_i) \cap U_i| \geq \lceil \sqrt{\Delta} \rceil$ and light otherwise. We let H and L denote the set of heavy and light vertices in S so that $|S| = |H| + |L|$.

To establish that $\lceil \sqrt{\Delta} \rceil$ -DOMINATE is $3\sqrt{\Delta}$ -competitive we use a charging argument, but it is quite different from the arguments in Sections 3.1 and 3.2. Initially, let $ch(v) = 1$ for each $v \in S$ so that $|S| = \sum_{v \in S} ch(v)$. Then spread the charge from S strictly to vertices in OPT so that $\sum_{v \in S} ch(v) = \sum_{v \in OPT} ch^*(v)$ where $ch^*(v)$ is the new charge on a vertex in OPT . We then show that $ch^*(v) \leq 2\sqrt{\Delta}$ for all $v \in OPT$ and thus $|S| = \sum_{v \in S} ch(v) =$

$\sum_{v \in OPT} ch^*(v) \leq |OPT|2\sqrt{\Delta}$ and the result then follows. We spread the charge from S to OPT according to the following rules:

1. If $v_i \in S \cap OPT$ then v_i keeps its full initial charge.
2. If $v_i \in H \setminus OPT$ then it spreads its initial charge evenly over all vertices in OPT . That is, each $v \in OPT$ obtains an additional charge of $\frac{1}{|OPT|}$ from v_i .
3. For each $v_i \in L \setminus OPT$, let $s(v_i)$ denote the set of vertices saved by v_i . Given a vertex $v_{i'} \in s(v_i)$ let opt be the mapping that maps $v_{i'}$ to itself if it is in OPT or to its earliest revealed neighbor in OPT otherwise. That is, $opt(v_{i'}) = v_{i'}$ if $v_{i'} \in OPT$ and $opt(v_{i'}) = v_k$ where $k = \min\{j \mid v_j \in N(v_{i'}) \cap OPT\}$ otherwise. For each $v_{i'} \in s(v_i)$, v_i spreads a charge of $\frac{1}{|s(v_i)|}$ to $opt(v_{i'})$.

► **Lemma 20.** If $v_i \in OPT$ then it receives charge from at most $\lceil \sqrt{\Delta} \rceil$ light vertices.

Proof. We consider two cases; (1) $v_i \in S$ or (2) $v_i \notin S$.

Case 1: Assume that $v_i \in S$, we show that v_i receives no charge from a distinct light vertex (therefore it receives charge from at most one light vertex, itself). Since $v_i \in S$ this implies that it is not saved by any $v_j, j \neq i$. Thus, if v_i were to receive charge from a light vertex it must be that $v_i = opt(v_{i'})$ for some $v_{i'}$ that is saved by some $v_k \in L$ different from v_i . More precisely, v_i must be adjacent to some $v_{i'}$ that is saved by some v_k with $k \neq i$. Yet, if $v_{i'} \in N(v_i)$ is saved then $N[v_{i'}] \cap S = \{v_i\}$ so this cannot be the case.

Case 2: Assume that $v_i \notin S$ and first remark that v_i is saved by at most one vertex so that it receives at most one charge from a light vertex in this way. If v_i receives charge from any other light vertex $v_k \in L$, it must be that v_i is adjacent to some vertex $v_{i'}$

that is saved by v_k . By Observation 5 it must be that $v_{i'} \in N(v_i) \cap U_i$, that is, is undominated when v_i is revealed. All this to say, that any light vertex that charges v_i determines at least one neighbor of v_i that is undominated at time i . Since $v_i \notin S$ we have $|N(v_i) \cap U_i| \leq \lceil \sqrt{\Delta} \rceil - 1$ and thus accounting for possibly one light vertex that charges v_i there are at most $\lceil \sqrt{\Delta} \rceil$ light vertices that charge v_i . ◀

► **Lemma 21.** $\frac{|H|}{|OPT|} \leq \sqrt{\Delta} + \frac{1}{\sqrt{\Delta}}$.

Proof. Since every vertex in H is selected because it dominated at least $\lceil \sqrt{\Delta} \rceil$ undominated vertices it follows that $|H| \leq \lfloor \frac{n}{\lceil \sqrt{\Delta} \rceil} \rfloor$. Moreover, by a standard result, first proved by Berge [1], a lower bound on OPT is $|OPT| \geq \lceil \frac{n}{\Delta+1} \rceil$. Ultimately this yields that $\frac{|H|}{|OPT|} \leq \frac{\lfloor \frac{n}{\lceil \sqrt{\Delta} \rceil} \rfloor}{\lceil \frac{n}{\Delta+1} \rceil} \leq \frac{\frac{n}{\lceil \sqrt{\Delta} \rceil}}{\frac{n}{\Delta+1}} \leq \frac{\Delta+1}{\sqrt{\Delta}} = \sqrt{\Delta} + \frac{1}{\sqrt{\Delta}}$. ◀

► **Theorem 22.** $\rho(\lceil \sqrt{\Delta} \rceil\text{-DOMINATE}, \Delta\text{-BOUNDED}) \leq 3\sqrt{\Delta}$.

Proof. Consider an arbitrary vertex $v_i \in OPT$. In light of Lemma 20 we see that it receives charge from at most $\lceil \sqrt{\Delta} \rceil$ light vertices, where each charge is no larger than 1. Moreover, by Lemma 21 the charge received by the heavy vertices is at most $\sqrt{\Delta} + \frac{1}{\sqrt{\Delta}}$ and v_i possibly receives charge from itself (it may be a heavy or light vertex). In particular we obtain that $ch(v_i) \leq \frac{|H|}{|OPT|} + \lceil \sqrt{\Delta} \rceil + 1 \leq (\sqrt{\Delta} + \frac{1}{\sqrt{\Delta}}) + \lceil \sqrt{\Delta} \rceil + 1 \leq 3\sqrt{\Delta}$. ◀

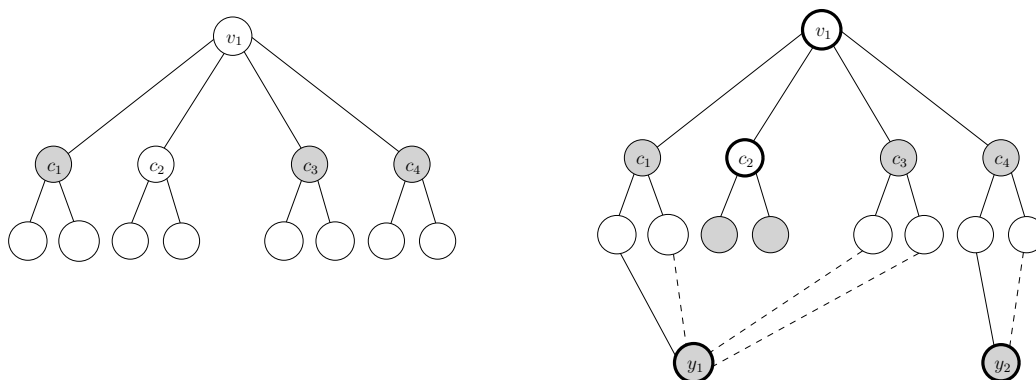
We now prove a lower bound $\Omega(\sqrt{\Delta})$ for any online algorithm. We should note that the adversarial input is bounded in size by a function of Δ . Although we have omitted the details, it is straightforward to extend the input so that the lower bound is in fact an asymptotic one.

► **Theorem 23.** $\rho(ALG, \Delta\text{-BOUNDED}) = \Omega(\sqrt{\Delta})$.

Proof. For simplicity we assume that Δ is a perfect square. Reveal v_1 with Δ children and reveal each child of v_1 with an additional $\sqrt{\Delta}$ children. Of the Δ children of v_1 , suppose that ALG selects exactly j where $0 \leq j \leq \Delta$. For the $\Delta - j$ vertices not selected, their $\sqrt{\Delta}$ neighbors are revealed to have degree 1 and ALG is forced to select each of these $(\Delta - j)(\sqrt{\Delta})$ vertices of degree 1.

Let S_j denote the set of the j selected vertices in $N(v_1)$ and $X = \bigcup_{v_i \in S_j} N(v_i)$. Since each vertex in S_j has $\sqrt{\Delta}$ children, it follows that $|X| = j\sqrt{\Delta}$. Partition the vertices of X into $\lceil \frac{j\sqrt{\Delta}}{\Delta} \rceil = \lceil \frac{j}{\sqrt{\Delta}} \rceil$ parts of size Δ (with at most one part having size $< \Delta$). Letting the parts be $X_1, X_2, \dots, X_{\lceil \frac{j}{\sqrt{\Delta}} \rceil}$ we reveal each vertex in a given part to a common vertex y_i (see figure 8 for an example). ALG must select at least one vertex for each part to dominate y_i and therefore at least an additional $\lceil \frac{j}{\sqrt{\Delta}} \rceil$ vertices are selected.

In total, ALG selects at least $j + (\Delta - j)(\sqrt{\Delta}) + \frac{j}{\sqrt{\Delta}}$ whereas OPT simply selects v_1 , the $\Delta - j$ vertices in $N(v_1) \setminus S_j$ and the $\frac{j}{\sqrt{\Delta}}$ vertices with labels y_i . Ultimately we have $\frac{ALG}{OPT} \geq \frac{j + (\Delta - j)(\sqrt{\Delta}) + \frac{j}{\sqrt{\Delta}}}{1 + (\Delta - j) + \frac{j}{\sqrt{\Delta}}} = \frac{j + j\sqrt{\Delta} + (\Delta - j)\Delta}{j + \sqrt{\Delta} + (\Delta - j)\sqrt{\Delta}} = \frac{\sqrt{\Delta}(j/\sqrt{\Delta} + j + (\Delta - j)\sqrt{\Delta})}{2(j/2 + \sqrt{\Delta}/2 + (\Delta - j)\sqrt{\Delta}/2)} \geq \frac{\sqrt{\Delta}}{2}$, where the last inequality follows from the fact that $j/2 + \sqrt{\Delta}/2 + (\Delta - j)\sqrt{\Delta}/2 \leq j/\sqrt{\Delta} + j + (\Delta - j)\sqrt{\Delta}$, since $\sqrt{\Delta}/2 \leq j/\sqrt{\Delta} + j/2 + (\Delta - j)\sqrt{\Delta}/2$, which can be seen since when $j < \Delta$ then the last term on the right hand side already is at least as large as the left hand side and when $j = \Delta$ then the middle term on the right hand side is at least the left hand side. ◀



■ **Figure 8** An instance described in the proof of Theorem 23 with $\Delta = 4$. The left depicts the graph after the children of v_1 have been revealed. Assuming that *ALG* selects $\{c_1, c_3, c_4\}$ above, the right depicts the completely revealed graph.

3.4 Graphs with Bounded Claws

In Appendix A we study $K_{1,t}$ -free graphs, which we also refer to as graphs with bounded “claws” (for $t = 3$, this graph class is known as “claw-free graphs”). We show that the competitive ratio $t - 1$ is both necessary and sufficient for this class of graphs. The upper bounds that we have demonstrated so far were all based on the k -DOMINATE algorithm for a suitable choice of parameter k . Interestingly, our upper bound on $K_{1,t}$ -free graphs is based on a conceptually simpler GREEDY algorithm. The analysis is no longer based on a charging scheme, but follows from combinatorial properties of graphs with bounded claws. For the details, one should consult the appendix.

4 Conclusions

In this paper we studied the minimum dominating set problem in an online setting where a vertex is revealed alongside all its neighbors. We also contrasted our results with those obtained by Boyar et al. [3] and Kobayashi [13] in a related vertex-arrival model. Dominating set is a difficult problem both offline and online. In our setting, the best achievable competitive ratio on general graphs is $O(\sqrt{n})$. This observation prompted us to study this problem with respect to more restrictive graph classes. Trees provide a natural graph class that usually allows for non-trivial competitive ratios. Indeed, we showed that in our model trees admit 2-competitive algorithms. There are several ways to try to extend this result to larger graph classes. We considered cactus graphs and showed that the optimal competitive ratio is 2.5 on them. Another way of generalizing trees is to consider graphs of higher treewidth. Unfortunately, once treewidth goes up to 2, competitive ratio jumps to $\Omega(\sqrt{n})$ (which is trivial in our setting due to $O(\sqrt{n})$ upper bound), as witnessed by series-parallel graphs. We also established non-trivial upper bounds on graphs of bounded degree, as well as graphs with bounded claws. When one moves to planar (even bipartite planar) graphs and threshold graphs, the competitive ratio jumps to $\Omega(\sqrt{n})$ again.

The above can be viewed as a larger program of developing a deeper understanding of the dominating set problem in an online setting. What are the main structural obstacles in graphs that prohibit online algorithms with small competitive ratios? Can one discover a family of graphs parameterized by some parameter t , which include cactus graphs, claw-free graphs, and bounded-degree graphs, such that the competitive ratio scales gracefully with t ? Lastly,

as another research direction, we mention that we have only considered the deterministic setting, so it would be of interest to extend our results to the randomized setting, as well as the setting of online algorithms with advice.

References

- 1 C. Berge. *The Theory of Graphs and Its Applications*. Methuen, 1962.
- 2 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- 3 Joan Boyar, Stephan J. Eidenbenz, Lene M. Favrholdt, Michal Kotrbčík, and Kim S. Larsen. Online dominating set. *Algorithmica*, 81(5):1938–1964, 2019.
- 4 Marek Cygan, Geevarghese Philip, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Dominating set is fixed parameter tractable in claw-free graphs. *Theoretical Computer Science*, 412(50):6982–7000, 2011.
- 5 Bevan Das and Vaduvur Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *1997 IEEE International Conference on Communications: Towards the Knowledge Millennium, ICC 1997, Montréal, Québec, Canada, June 8–12, 1997*, pages 376–380. IEEE, 1997.
- 6 Hovhannes Harutyunyan, Denis Pankratov, and Jesse Raciocot. Online domination: The value of getting to know all your neighbors, 2021. [arXiv:2105.00299](https://arxiv.org/abs/2105.00299).
- 7 Hovhannes A. Harutyunyan. An efficient vertex addition method for broadcast networks. *Internet Math.*, 5(3):211–225, 2008.
- 8 Hovhannes A. Harutyunyan and Arthur L. Liestman. Upper bounds on the broadcast function using minimum dominating sets. *Discret. Math.*, 312(20):2992–2996, 2012.
- 9 T.W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. Marcel Dekker, New York, 1998.
- 10 Stephen T. Hedetniemi, Renu C. Laskar, and John Pfaff. A linear algorithm for finding a minimum dominating set in a cactus. *Discret. Appl. Math.*, 13(2-3):287–292, 1986.
- 11 M. Henning and A. Yeo. *Total Domination in Graphs*. Springer-Verlag New York, 2013.
- 12 Gow-Hsing King and Wen-Guey Tzeng. On-line algorithms for the dominating set problem. *Inf. Process. Lett.*, 61(1):11–14, 1997.
- 13 Koji M. Kobayashi. Improved bounds for online dominating sets of trees. In Yoshio Okamoto and Takeshi Tokuyama, editors, *28th International Symposium on Algorithms and Computation, ISAAC 2017, December 9–12, 2017, Phuket, Thailand*, volume 92 of *LIPICs*, pages 52:1–52:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 14 Dennis Komm. *An Introduction to Online Computation – Determinism, Randomization, Advice*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016.
- 15 D. König. *Theorie der Endlichen und Unendlichen Graphen*. Chelsea, New York, 1950.
- 16 O. Ore. *Theory of Graphs*. American Mathematical Society, 1962.
- 17 Feng Wang, Ding-Zhu Du, and Xiuzhen Cheng. Connected dominating set, 2016. In *Encyclopedia of Algorithms*.

A Tight Bound for Graphs with Bounded Claws

Let $t \geq 3$, a graph G is said to be $K_{1,t}$ -free if it contains no induced subgraph isomorphic to $K_{1,t}$. When $t = 3$, this is the well-studied class of claw-free graphs. In this section we study $K_{1,t}$ -free graphs, which we also refer to as graphs with bounded “claws”.

From the preceding sections one might notice that the existence of an induced subgraph $K_{1,t}$ poses challenges for an algorithm. This section suggests that this intuition holds more than just a grain of truth. We show that, when inputs are restricted to $K_{1,t}$ -free graphs, the competitive ratio of every algorithm is bounded below by $t - 1$ and there is an algorithm that

achieves competitive ratio $t - 1$. The upper bounds that we have demonstrated so far were all based on the k -DOMINATE algorithm for a suitable choice of parameter k . Interestingly, our upper bound on $K_{1,t}$ -free graphs is based on a conceptually simpler GREEDY algorithm. The analysis is no longer based on a charging scheme, but follows from combinatorial properties of graphs with bounded claws.

► **Theorem 24.** $\rho(\text{ALG}, K_{1,t}\text{-FREE}) \geq t - 1$.

Proof. Reveal v_1 with $t - 1$ children. If ALG does not select v_1 then the input terminates as a star on t vertices (i.e. the $t - 1$ neighbors of v_1 are revealed with no additional neighbors). Any feasible algorithm must select the $t - 1$ neighbors of v_1 whereas OPT selects v_1 and the statement then follows. Suppose that ALG selects v_1 and let $c_i, 1 \leq i \leq t - 1$ be the children of v_1 . Reveal c_1 as adjacent to each child of v_1 and with an additional $t - 2$ children. If ALG does not select c_1 then the children of c_1 are revealed as leaves whereas the rest of the input is revealed to be a clique. That is, $N[v_1]$ is a clique and only c_1 has children. ALG selected v_1 and is forced to select the $t - 2$ children of c_1 whereas OPT selects only c_1 as a single dominating vertex. It is not hard to see that this input is $K_{1,t}$ -free and the result then follows (see Figure 9 for an example).

Suppose that ALG selects c_1 , the input then continues in the following way; For each $2 \leq j \leq t - 2$, (as long as ALG is accepting c_j) we reveal c_j as adjacent to every visible vertex and with an additional $t - 3$ children. That is, c_j is adjacent to each child $c_i, i \neq j$ of v_1 and the grandchildren of v_1 (i.e. the children of all the c_i with $1 \leq i \leq j$) so that c_j is a single dominating vertex of this prefix.

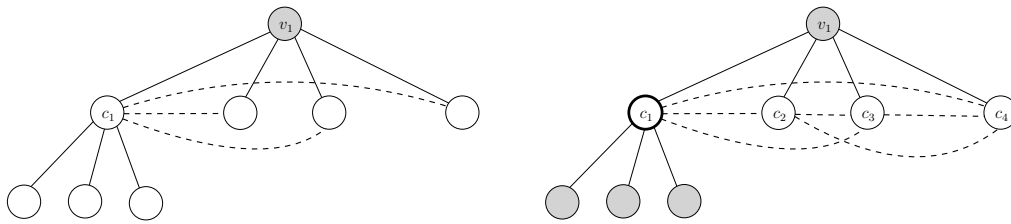
Case 1: If there is some $2 \leq j \leq t - 2$ such that ALG does not select c_j then the $t - 3$ children of

c_j are revealed as leaves, $N[v_i]$ is revealed as a clique, and the $(t - 2) + \sum_{i=2}^j (t - 3) = j(t - 3) + 1$ grandchildren of v_1 are revealed to form a clique. At this point, ALG has selected $\{v_1, c_1, \dots, c_{j-1}\}$ and is now forced to select the $t - 3$ children of c_j for an output of at least $j + (t - 3) \geq 2 + (t - 3) = t - 1$ whereas OPT selects only c_j so that $\frac{\text{ALG}}{\text{OPT}} \geq \frac{t-1}{1}$.

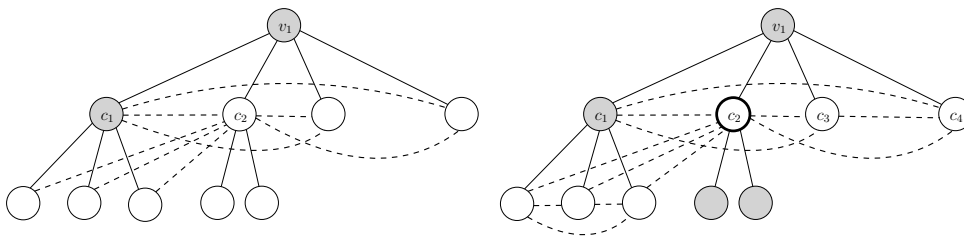
We now argue that this input is $K_{1,t}$ -free. Notice that for all v in this input we have $N(v) \subseteq N(c_j)$ so that if there is an induced $K_{1,t}$ with central vertex v then there is a claw with central vertex c_j . Therefore it is sufficient to show that is no claw with central vertex c_j to finish the claim. Suppose for contradiction's sake that there were an induced $K_{1,t}$ where c_j is the central vertex and the t neighbors of c_j are all pairwise non-adjacent. Let G denote the grandchildren of v_1 and remark that any neighbor of c_j is either a child of c_j , a grandchild of v_1 , or a vertex from $N[v_1] \setminus \{c_j\}$. Since there are t vertices and c_j only has $t - 3$ children by the pigeonhole principle we must have at least two vertices u, v that both are grandchildren of v_1 or both belong $N[v_1] \setminus \{c_j\}$. Yet, both the set of grandchildren of v_1 and $N[v_1] \setminus \{c_j\}$ are cliques. Therefore we have that u and v are adjacent, contradicting our assumption.

Case 2: If ALG selects each $c_i, 1 \leq i \leq t - 2$ then the $(t - 2)(t - 3) + 1$ grandchildren of v_1 are then revealed to form a clique ($N[v_1]$ has already been revealed as a clique). ALG has already selected $\{v_1, c_1, \dots, c_{t-2}\}$ and therefore has an output of at least $t - 1$ whereas OPT selects only c_{t-2} . An argument similar to the one above will yield that this input is $K_{1,t}$ -free and the result then follows. ◀

When inputs are restricted to $K_{1,t}$ -free graphs, we show that the online algorithm GREEDY is $(t - 1)$ -competitive. The crucial observation to make here is that the output of GREEDY is an independent set. We provide a result below that is a straightforward generalization of one given in [4]. The simplicity of the result suggests that it may have appeared in earlier work.



■ **Figure 9** An instance described in Theorem 24 with $t = 5$ where ALG does not select c_1 . The left depicts the graph at the moment c_1 was revealed and the right depicts the completely revealed graph.



■ **Figure 10** An instance described in Theorem 24 with $t = 5$ where ALG does not select c_2 . The left depicts the graph at the moment c_1 was revealed and the right depicts the completely revealed graph.

► **Lemma 25.** *Let $t \geq 3$, $G = (V, E)$ be a $K_{1,t}$ -free graph and I be any independent set in G . Then $|D| \geq \frac{|I|}{t-1}$ for any dominating set D in G .*

Proof. Suppose for the sake of deriving a contradiction that there is some dominating set D in G with $|D| < \frac{|I|}{t-1}$. Remarking that the vertices of D dominate the vertices of I as D is a dominating set we notice that there is some vertex $v \in D$ that dominates at least t vertices of I (i.e. if every vertex of D dominated at most $t - 1$ vertices then D would dominate at most $(t - 1)|D| < |I|$ vertices). Moreover, since v is adjacent to at least one of the $t \geq 3$ vertices of I it dominates, it cannot belong to I as I is independent. Therefore, the vertices of I dominated by $v \notin I$ are adjacent to v . In particular, at least t vertices of I , all pairwise non-adjacent, are neighbors of v and this induces $K_{1,t}$ in G . ◀

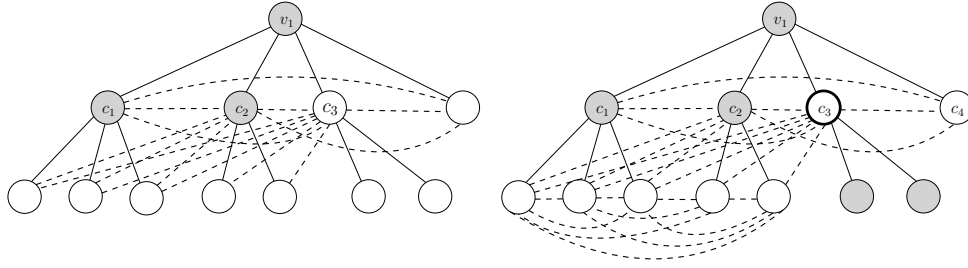
The preceding lemma shows that for any independent set I in a $K_{1,t}$ -free graph G , $|I| \leq (t - 1)\gamma(G)$. Given that GREEDY outputs an independent set we obtain the following result which is of interest to us.

► **Theorem 26.** $\rho(\text{GREEDY}, K_{1,t}\text{-FREE}) = t - 1$.

Proof. The upper bound is a consequence of Proposition 25 and the remarks that follow. The lower bound follows from Theorem 24. ◀

B Noncompetitive Graph Classes

Recall that the setting defined in [12] is nearly identical to ours except that an algorithm knows the input size n beforehand and the induced subgraph on the revealed vertices is not necessarily connected. Within this setting the authors establish a lower bound of $\Omega(\sqrt{n})$ for arbitrary graphs. Their proof can be augmented to show a lower bound of $\Omega(\sqrt{n})$ in our model, which is tight by our upper bound of $O(\sqrt{\Delta})$ on degree at most Δ graphs (applied



■ **Figure 11** An instance described in Theorem 24 with $t = 5$ where ALG does not select c_3 . The left depicts the graph at the moment c_3 was revealed and the right depicts the completely revealed graph.

to $\Delta = n - 1$). Instead, we strengthen such a result in several ways by showing that the lower bound of $\Omega(\sqrt{n})$ applies to several restricted classes such as threshold graphs⁴, planar bipartite graphs, and series-parallel graphs. Some of the proofs are omitted in this section due to space limitations. These proofs can be found in the full version of the paper [6].

B.1 Threshold Graphs

The graph join operation applied to two graphs G_1 and G_2 takes the disjoint union of the two graphs and adds all possible edges between the two graphs to the result (in addition to retaining the edges of G_1 and G_2). The class of threshold graphs can be described recursively as follows:

1. K_1 (i.e. a single isolated vertex) is a threshold graph.
2. If G is a threshold graph then the disjoint union $G \cup K_1$ is a threshold graph.
3. If G is a threshold graph then the graph join $G \oplus K_1$ is a threshold graph.

It is not hard to see that any connected threshold graph has a dominating set of size 1. Since our setting only allows for connected graphs we instead measure ALG as a function of input size n since $OPT \leq 1$ on every input. In particular, we show that for any algorithm there is an infinite family of threshold graphs for which this algorithm selects $\Omega(\sqrt{n})$ vertices (where the input has n vertices). Although OPT does not tend towards infinity, we consider this to be an asymptotic lower bound, but with input size n tending to infinity. In a sense, this is a stronger lower bound since the algorithm is guaranteed an input graph with a single dominating vertex, yet it still selects more than $\Omega(\sqrt{n})$ vertices in the input.

► **Theorem 27.** *For infinitely many values of n there is a threshold graph G_n such that $ALG(G_n) = \Omega(\sqrt{n})$.*

B.2 Planar Bipartite Graphs

Below is a lower bound of $\Omega(\sqrt{n})$ for planar bipartite graphs. We should mention that is strikingly similar to the lower bound on general graphs given in [12]. We provide a simple augmentation of their lower bound so that it not only consists of inputs that are revealed according to our model but inputs that are also planar bipartite graphs.

► **Theorem 28.** $\rho(ALG, PLANAR\ BIPARTITE) = \Omega(\sqrt{n})$.

⁴ With the caveat that, for threshold graphs, we instead consider the performance ratio as a function of input size.

B.3 Series-Parallel Graphs

In light of our 2-competitive algorithm for trees, it is natural to suppose that some class of graphs generalizing trees might admit competitive algorithms, that is, algorithms with bounded competitive ratio. One such generalization is graphs of bounded treewidth. Trees have treewidth 1, so the next step is to consider graphs of treewidth 2. Unfortunately, in this section we show that by increasing treewidth parameter from 1 to 2, the online dominating set problem becomes extremely hard for online algorithms. More specifically, we show that series-parallel graphs do not admit online algorithms with competitive ratio better than $\Omega(\sqrt{n})$. We remark that series-parallel graphs have treewidth at most 2.

We begin by recalling the definition of a series-parallel graph. It is defined with the help of the notion of a two-terminal graph (G, s, t) , which is a graph G with two distinguished vertices s , called a source, and t , called a sink. For a pair of two-terminal graphs (G_1, s_1, t_1) and (G_2, s_2, t_2) , there are two composition operations:

- *Parallel composition*: take a disjoint union of G_1 with G_2 and merge s_1 with s_2 to get the new source, as well as t_1 with t_2 to get the new sink.
- *Series composition*: take a disjoint union of G_1 with G_2 and merge t_1 with s_2 , which now becomes an inner vertex of the resulting two-terminal graph; s_1 becomes the new source and t_2 becomes the new sink.

A two-terminal series-parallel graph is a two-terminal graph that can be obtained by starting with several copies of the K_2 graph and applying a sequence of parallel and series compositions. Lastly, a graph is called series-parallel if it is a two-terminal series-parallel graph for some choice of source and sink vertices. Observe that intermediate graphs resulting in the construction of a series-parallel graph may have multiple parallel edges, so they are multigraphs. This is permitted, as long as the resulting overall graph is a simple undirected graph at the end.

Now, we are ready to prove the main result of this section.

► **Theorem 29.** $\rho(\text{ALG}, \text{SERIES-PARALLEL}) = \Omega(\sqrt{n})$.

Proof. Let $k \geq 2$ be an integer. The adversary reveals s with k neighbors c_1, \dots, c_k . Then c_1, \dots, c_k are revealed in this order with k new neighbors each. Let neighbors of c_i be d_{i1}, \dots, d_{ik} . Let $S \subseteq \{c_1, \dots, c_k\}$ be those vertices selected by *ALG*. For those $i \notin S$ we reveal their new neighbors in order d_{i1}, \dots, d_{ik} . Each such d_{ij} is revealed with a single new neighbor f_{ij} . For $i \in S$ we reveal their new neighbors in order d_{i1}, \dots, d_{ik} . Each such d_{ij} is revealed with a new neighbor t that is common to all these vertices. Then f_{ij} are revealed in arbitrary order with t as a new neighbor. Lastly t is revealed without any new neighbors.

Let $p = |S|$. Observe that in addition to these p vertices *ALG* must select at least one vertex from each of $\{d_{ij}, f_{ij}\}$ pairs for those $i \notin S$; otherwise, vertex d_{ij} would be undominated. Thus, $\text{ALG} \geq p + k(k - p)$. Also, observe that $\{s, t\} \cup \{c_i \mid i \notin S\}$ is a dominating set, so $\text{OPT} \leq k - p + 2$. The bound on the competitive ratio is $\frac{\text{ALG}}{\text{OPT}} \geq \frac{p + k(k - p)}{k - p + 2} = k - \frac{2k - p}{k - p + 2} \geq \frac{k}{2}$, where the last inequality is obtained as follows. For $k \geq 2$ we have $k^2 - kp \geq 2k - 2p$, which implies $k^2 - kp + 2k \geq 4k - 2p$. This in turn implies that $k(k - p + 2) \geq 2(2k - p)$, hence $(2k - p)/(k - p + 2) \leq k/2$. The quantitative part of the statement of this theorem follows from the fact that the total number of vertices is at most $2 + k + k^2 + k(k - p) = \Theta(k^2)$.

Lastly, we note that the adversarial graph thus constructed is, indeed, series-parallel. For each $i \notin S$ and $j \in \{1, \dots, k\}$ the path $c_i \rightarrow d_{ij} \rightarrow f_{ij} \rightarrow t$ is a series-composition of 3 copies of K_2 . These paths can be merged by a parallel composition to obtain the subgraph induced on $\{c_i, t\} \cup \{d_{ij}, f_{ij} \mid j \in \{1, \dots, k\}\}$ for each $i \notin S$. Each of these subgraphs is composed at c_i with another copy of K_2 with the new vertex playing the role of s . Similar argument holds to show that the subgraph induced on $\{s, c_i, t\} \cup \{d_{ij} \mid j \in \{1, \dots, k\}\}$ for $i \in S$ is a two-terminal series-parallel graph. Lastly, all these subgraphs are merged by a sequence of parallel compositions at s and t . ◀

A Linear-Time Nominal μ -Calculus with Name Allocation

Daniel Hausmann  

Gothenburg University, Göteborg, Sweden

Stefan Milius  

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Lutz Schröder  

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Logics and automata models for languages over infinite alphabets, such as Freeze LTL and register automata, serve the verification of processes or documents with data. They relate tightly to formalisms over nominal sets, such as nondeterministic orbit-finite automata (NOFAs), where names play the role of data. Reasoning problems in such formalisms tend to be computationally hard. *Name-binding* nominal automata models such as *regular nondeterministic nominal automata (RNNAs)* have been shown to be computationally more tractable. In the present paper, we introduce a linear-time fixpoint logic $\text{Bar-}\mu\text{TL}$ for finite words over an infinite alphabet, which features full negation and freeze quantification via name binding. We show by a nontrivial reduction to *extended regular nondeterministic nominal automata* that even though $\text{Bar-}\mu\text{TL}$ allows unrestricted nondeterminism and unboundedly many registers, model checking $\text{Bar-}\mu\text{TL}$ over RNNAs and satisfiability checking both have elementary complexity. For example, model checking is in 2EXPSPACE , more precisely in parametrized EXPSPACE , effectively with the number of registers as the parameter.

2012 ACM Subject Classification Theory of computation \rightarrow Modal and temporal logics; Theory of computation \rightarrow Verification by model checking

Keywords and phrases Model checking, linear-time logic, nominal sets

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.58

Related Version *Extended Version*: <https://arxiv.org/abs/2010.10912>

Funding *Daniel Hausmann*: Supported by Deutsche Forschungsgemeinschaft (DFG) under project MI 717/7-1 and by the European Research Council (ERC) under project 772459.

Stefan Milius: Supported by Deutsche Forschungsgemeinschaft (DFG) under project MI 717/7-1.

Lutz Schröder: Supported by Deutsche Forschungsgemeinschaft (DFG) under project SCHR 1118/15-1.

1 Introduction

There has been longstanding interest in logics and automata models over infinite alphabets, such as the classical register automaton model [24] and Freeze LTL (e.g. [11,31]), or automata models over nominal sets [36] such as nondeterministic orbit-finite automata [2], in which *names* play the role of letters. Infinite alphabets may be seen as representing data. For example, nonces in cryptographic protocols [30], data values in XML documents [35], object identities [18], or parameters of method calls [23] can all be usefully understood as letters in infinite alphabets. A central challenge in dealing with infinite alphabets is that key decision problems in many logics and automata models are either undecidable or of prohibitively high complexity unless drastic restrictions are imposed (see the related work section). In a nutshell, the contribution of the present work is the identification of a linear-time fixpoint logic $\text{Bar-}\mu\text{TL}$ for *finite* words over infinite alphabets that



© Daniel Hausmann, Stefan Milius, and Lutz Schröder;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 58; pp. 58:1–58:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- allows both safety and liveness constraints (via fixpoints, thus in particular going beyond the expressiveness of LTL [17]) as well as full nondeterminism, and e.g. expresses the language “some letter occurs twice” (which cannot be accepted by deterministic or unambiguous register automata [2, 37]);
- imposes no restriction on the number of registers; and
- is closed under complement;

and nevertheless allows model checking and satisfiability checking in elementary complexity: Bar- μ TL model checking over regular nondeterministic nominal automata [38] (in the sense of checking that all words accepted by a given automaton satisfy a given formula) is in 2EXPSpace and more precisely in parametrized EXPSpace, with the maximal size of the support of states as the parameter (in the translation of nominal automata to register automata [2, 38], this corresponds to the number of registers); and satisfiability checking is in EXPSpace (and in parametrized PSPACE).

The tradeoff that buys this comparatively low complexity is a mild recalibration of the notion of freshness, which we base on explicit binding of names in strings in a nominal language model; depending on the exact discipline of α -renaming of bound names, one obtains either global freshness (w.r.t. all previous letters, as in session automata [3]) or local freshness (w.r.t. currently stored letters, as in register automata). This principle has been previously employed in the semantics of nominal Kleene algebra [16, 28] and in regular non-deterministic nominal automata (RNNAs) [38]. It carries the limitation that in the local freshness variant, letters can be required to be distinct only from such previous letters that are expected to be seen again – a restriction that seems reasonable in applications; that is, in many situations, one presumably would not need to insist on an object identifier, nonce, or process name b to be distinct from a previous name a if a is never going to be used again.

We introduce a dedicated finite-word automaton model, *extended regular nondeterministic nominal automata (ERNNAs)*, which extend RNNAs by \top -states, i.e. deadlocked universal states. We then base our results on mutual translations between Bar- μ TL and ERNNAs; the tableau-style logic-to-automata translation turns out to be quite nontrivial as it needs to avoid the accumulation of renamed copies of subformulae in automata states. Since RNNAs are, under global freshness, essentially equivalent to session automata [3], one consequence of our results is that session automata (which as such are only closed under *resource-bounded* complementation [3]) can be made closed under complement simply by allowing \top -states.

Proofs are mostly omitted or only sketched; full proofs are in the extended version [22].

Related work. *Automata:* Over infinite alphabets, the expressive power of automata models generally increases with the power of control (deterministic/nondeterministic/alternating) [25]. In deterministic models, language inclusion can often be decided in reasonable complexity; this remains true for unambiguous register automata [5, 34]. For nondeterministic register automata and the equivalent nondeterministic orbit-finite automata [2], emptiness is decidable but inclusion is undecidable unless one restricts to at most two registers [24]. Similarly, language inclusion (equivalently nonemptiness) of alternating register automata is undecidable unless one restricts to at most one register, and even then is not primitive recursive [11]. Automata models for infinite words outside the register paradigm include data walking automata [33], whose inclusion problem is decidable even under nondeterminism but at least as hard as reachability in Petri nets (equivalently vector addition systems) [7], as well as the highly expressive data automata [1], whose nonemptiness problem is decidable but, again, at least as hard as Petri net reachability. Note that by recent results [9], Petri net reachability is not elementary, and in fact Ackermann-complete [10, 32].

Logics: Bar- μ TL is incomparable to Freeze LTL. Satisfiability checking of Freeze LTL is by reduction to alternating register automata, and has the same high complexity even for the one-register case [11]. Satisfiability in the *safety* fragment of Freeze LTL over infinite words [31] is EXPSPACE-complete if the number of registers is restricted to at most one, while the refinement and model checking problems are decidable but not primitive recursive (all three problems become undecidable in presence of more than one register). The μ -calculus *with atoms* [26] is interpreted over Kripke models with atoms; its satisfiability problem is undecidable while its model checking problem is decidable, with the complexity analysis currently remaining open. It is related to the very expressive *first-order μ -calculus* [19, 20], for which model checking is only known to be semidecidable. The μ -calculus *over data words* [6, 8] works in the data walking paradigm. The satisfiability problem of the full calculus is undecidable; that of its ν -fragment, which translates into data automata, is decidable but elementarily equivalent to Petri net reachability. *Variable LTL* [21] extends LTL with a form of first-order quantification over data domains. The full language is very expressive and in particular contains full Freeze LTL [39]. Some fragments have decidable satisfiability or model checking problems (typically not both) [21, 39], occasionally in elementary complexity; these impose prenex normal form (reducing expressiveness) and restrict to quantifier prefixes that fail to be stable under negation. Decidable fragments will, of course, no longer contain full Freeze LTL; how their expressiveness compares to Bar- μ TL needs to be left open at present. *Flat Freeze LTL* [12] interdicts usage of the freeze quantifier in safety positions (e.g. the freeze quantifier can occur under F but not under G); its existential model checking problem over (infinite runs of) one-counter automata is NEXPTIME-complete, while the universal model checking problem is undecidable [4].

2 Preliminaries: Nominal Sets

Nominal sets offer a convenient formalism for dealing with names and freshness; for our present purposes, names play the role of data. We briefly recall basic notions and facts (see [36] for more details).

Fix a countably infinite set \mathbb{A} of *names*, and let G denote the group of finite permutations on \mathbb{A} , which is generated by the *transpositions* (ab) for $a \neq b \in \mathbb{A}$ (recall that (ab) just swaps a and b); we write 1 for the neutral element of G . A (left) *action* of G on a set X is a map $(-) \cdot (-): G \times X \rightarrow X$ such that $1 \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi\pi') \cdot x$ for all $x \in X$, $\pi, \pi' \in G$. For instance, G acts on \mathbb{A} by $\pi \cdot a = \pi(a)$. A set $S \subseteq \mathbb{A}$ is a *support* of $x \in X$ if $\pi(x) = x$ for all $\pi \in G$ such that $\pi(a) = a$ for all $a \in S$. We say that x is *finitely supported* if x has some finite support, and *equivariant* if the empty set is a support of x , i.e. $\pi \cdot x = x$ for all $\pi \in G$. The *orbit* of $x \in X$ is the set $\{\pi \cdot x \mid \pi \in G\}$. The set of all orbits forms a partition of X , and X is *orbit-finite* if there are finitely many orbits.

A *nominal set* is a set X equipped with an action of G such that every element of X is finitely supported; e.g. \mathbb{A} itself is a nominal set. An early motivating example of a nominal set is the set of λ -terms with variable names taken from \mathbb{A} , and with the action of $\pi \in G$ given by replacing every variable name a with $\pi(a)$ as expected, e.g. $(ab) \cdot \lambda a.ba = \lambda b.ab$. Every element x of a nominal set has a least finite support, denoted $\text{supp}(x)$, which one may roughly think of as the set of names occurring (“freely”, see below) in x . A name $a \in \mathbb{A}$ is *fresh* for x if $a \notin \text{supp}(x)$. On subsets $A \subseteq X$ of a nominal set X , G acts by $\pi \cdot A = \{\pi \cdot x \mid x \in A\}$. Thus, $A \subseteq X$ is equivariant iff $\pi \cdot A \subseteq A$ for all $\pi \in G$. Similarly, A has support S iff $\pi \cdot A \subseteq A$ whenever $\pi(a) = a$ for all $a \in S$. We say that A is *uniformly finitely supported* if $\bigcup_{x \in A} \text{supp}(x)$ is finite [40], in which case A is also finitely supported [14, Thm. 2.29].

(The converse does not hold, e.g. the set \mathbb{A} is finitely supported but not uniformly finitely supported.) Uniformly finitely supported subsets of orbit-finite sets are always finite but in general, uniformly finitely supported sets can be infinite; e.g. for finite $B \subseteq \mathbb{A}$, the set $B^* \subseteq \mathbb{A}^*$ is uniformly finitely supported.

The Cartesian product $X \times Y$ of nominal sets X, Y is a nominal set under the componentwise group action; then, $\text{supp}(x, y) = \text{supp}(x) \cup \text{supp}(y)$. Given a nominal set X equipped with an equivalence relation \sim that is equivariant as a subset of $X \times X$, the quotient X/\sim is a nominal set under the group action $\pi \cdot [x]_{\sim} = [\pi \cdot x]_{\sim}$. A key role in the technical development is played by *abstraction sets*, which provide a semantics for binding mechanisms [15]:

► **Definition 2.1** (Abstraction set). Given a nominal set X , an equivariant equivalence relation \sim on $\mathbb{A} \times X$ is defined by

$$(a, x) \sim (b, y) \quad \text{iff} \quad (ac) \cdot x = (bc) \cdot y \text{ for some } c \in \mathbb{A} \text{ that is fresh for } (a, x, b, y)$$

(equivalently for all such c). The *abstraction set* $[\mathbb{A}]X$ is the quotient set $(\mathbb{A} \times X)/\sim$. The \sim -equivalence class of $(a, x) \in \mathbb{A} \times X$ is denoted by $\langle a \rangle x \in [\mathbb{A}]X$.

We may think of \sim as an abstract notion of α -equivalence, and of $\langle a \rangle$ as binding the name a . Indeed we have $\text{supp}(\langle a \rangle x) = \text{supp}(x) \setminus \{a\}$, as expected in binding constructs.

3 Data Languages and Bar Languages

As indicated, we use the set \mathbb{A} of names as the data domain, and capture freshness of data values via α -equivalence, roughly as follows. We work with words over \mathbb{A} where names may be preceded by the bar symbol “|”, which indicates that the next letter is bound until the end of the word (cf. Remark 3.2); such words are called *bar strings* [38]. Bar strings may be seen as patterns that govern how letters are read from the input word; broadly speaking, an occurrence of la corresponds to reading a letter from the input word, and binding this letter to the name a , while an undecorated occurrence of a means that the letter referred to by a occurs in the input word. (We loosely speak of the input word as consisting of letters, and of bar strings as consisting of names; formally, however, letters and names are the same, viz., elements of \mathbb{A} .) Bound names can be renamed, giving rise to a notion of α -equivalence; as usual, the new name needs to be sufficiently *fresh*, i.e. cannot already occur freely in the scope of the binding. For instance, in $albab$, the $|$ binds the letter b in $lbab$. The bar string $albab$ is α -equivalent to $alcac$ but not, of course, to $alaaa$. That is, we can rename the bound name b into c but not into a , as a already occurs freely in $lbab$; we say that renaming b into a is *blocked*.

We will see that bar strings modulo α -equivalence relate to formalisms for *global* freshness (a name is globally fresh if it has never been seen before), such as session automata [3]. Contrastingly, if we regard a bar string as representing all words over \mathbb{A} that arise by performing some α -equivalent renaming and then removing the bars, we arrive at a notion of *local* freshness, similar to freshness w.r.t. currently stored names as in register automata; precise definitions are given later in this section. For instance, the bar string $alba$ represents the set of words $\{cdc \mid c, d \in \mathbb{A}, c \neq d\} \subseteq \mathbb{A}^*$ under both local and global freshness semantics – in $alba$, a and b cannot be renamed into the same letter, since a occurs freely in the scope of lb . Contrastingly, lab represents the set $\{cd \mid c \neq d\} \subseteq \mathbb{A}^*$ under global freshness semantics, but under local freshness semantics it just represents the set \mathbb{A}^2 of all two-letter words, since lab is α -equivalent to $lala$. The impossibility of expressing the language $\{cd \mid c \neq d\}$ under local freshness is thus hardwired into our language model. We emphasize again that this

restriction seems reasonable in practice, since one may expect that freshness of new letters is often relevant only w.r.t. letters that are intended to be seen again later, e.g. in deallocation statements or message acknowledgements. Formal definitions are as follows.

► **Definition 3.1** (Bar strings). We put $\bar{\mathbb{A}} = \mathbb{A} \cup \{la \mid a \in \mathbb{A}\}$; we refer to elements $la \in \bar{\mathbb{A}}$ as *bar names*, and to elements $a \in \mathbb{A}$ as *plain names*. A *bar string* is a word $w = \sigma_1\sigma_2 \cdots \sigma_n \in \bar{\mathbb{A}}^*$, with *length* $|w| = n$; we denote the empty string by ϵ . We turn $\bar{\mathbb{A}}$ into a nominal set by putting $\pi \cdot a = \pi(a)$ and $\pi \cdot la = l\pi(a)$; then, $\bar{\mathbb{A}}^*$ is a nominal set under the pointwise action of G . We define α -*equivalence* on bar strings to be the least equivalence \equiv_α such that

$$wlav \equiv_\alpha wlbv \quad \text{whenever} \quad \langle a \rangle v = \langle b \rangle u \text{ in } [\mathbb{A}] \bar{\mathbb{A}}^*$$

(Definition 2.1) for $w, v, u \in \bar{\mathbb{A}}^*$, $a \in \mathbb{A}$. Thus, la binds a , with scope extending to the end of the word. Correspondingly, a name a is *free* in a bar string w if there is an occurrence of a in w that is to the left of any occurrence of la . We write $[w]_\alpha$ for the α -equivalence class of $w \in \bar{\mathbb{A}}^*$ and $\text{FN}(w) = \{a \in \mathbb{A} \mid a \text{ is free in } w\}$ ($= \text{supp}([w]_\alpha)$) for the set of free names of w . If $\text{FN}(w) = \emptyset$, then w is *closed*. A bar string w is *clean* if all bar names la in w are pairwise distinct and have $a \notin \text{FN}(w)$. For a set $S \subseteq \mathbb{A}$ of names, we write $\text{bs}(S) = \{w \in \bar{\mathbb{A}}^* \mid \text{FN}(w) \subseteq S\}$.

► **Remark 3.2.** Closed bar strings are essentially the same as the *well-formed symbolic words* that appear in the analysis of session automata [3]. Indeed, symbolic words consist of operations that read a letter into a register, corresponding to bar names, and operations that require seeing the content of some register in the input, corresponding to plain names. Symbolic words are normalized by a register allocation procedure similar to α -renaming. Well-formedness of symbolic words corresponds to closedness of bar strings.

Moreover, modulo the respective equational laws, bar strings coincide with the ν -strings [28, 29] that appear in the semantics of *Nominal Kleene Algebra (NKA)* [16]; cf. [38]. These are constructed from names in \mathbb{A} , sequential composition, and a binding construct $\nu a. w$, which binds the name a in the word w . In particular, the equational laws of ν -strings allow extruding the scope of every ν to the end of the word after suitable α -renaming. We note that **Bar- μ TL** and its associated automata models are more expressive than NKA as they express languages with unbounded nesting of binders [38].

We will work with three different types of languages:

► **Definition 3.3.**

1. *Data languages* are subsets of \mathbb{A}^* .
2. *Literal languages* are subsets of $\bar{\mathbb{A}}^*$, i.e. sets of bar strings.
3. *Bar languages* are subsets of $\bar{\mathbb{A}}^* / \equiv_\alpha$, i.e. sets of α -equivalence classes of bar strings.

A bar language L is *closed* if $\text{supp}(L) = \emptyset$.

Bar languages are the natural semantic domain of our formalisms, and relate tightly to data languages as discussed next. A key factor in the good computational properties of regular nominal nondeterministic automata (RNNA) [38] is that the bar languages they accept (cf. Section 5) are uniformly finitely supported, and we will design **Bar- μ TL** to ensure the same property. Note that a uniformly supported bar language is closed iff it consists of (equivalence classes of) closed bar strings. For brevity, we will focus the exposition on target formulae (in model checking) and automata that denote or accept, respectively, closed bar languages, with free names appearing only in languages accepted by non-initial states or denoted by proper subformulae of the target formula. (The treatment is easily extended to

bar languages with free names; indeed, such globally free names are best seen as a separate finite alphabet of constant symbols [29].) We will occasionally describe example bar languages as regular expressions over $\bar{\mathbb{A}}$ (i.e. as *regular bar expressions* [38]), meaning the set of all α -equivalence classes of instances of the expression.

To convert bar strings into data words, we define $\mathbf{ub}(a) = \mathbf{ub}(|a) = a$ and extend \mathbf{ub} to bar strings letterwise; i.e. $\mathbf{ub}(w)$ is the data word obtained by erasing all bars “|” from w . We then define two ways to convert a bar language L into a data language:

$$N(L) = \{\mathbf{ub}(w) \mid [w]_\alpha \in L, w \text{ clean}\} \quad \text{and} \quad D(L) = \{\mathbf{ub}(w) \mid [w]_\alpha \in L\}.$$

That is, N is a global freshness interpretation of $|$, while D provides a local freshness interpretation as exemplified above; e.g. as indicated above we have $D(|alba) = N(|alba) = \{aba \mid a, b \in \mathbb{A}, a \neq b\}$, while $N(|alb) = \{ab \mid a, b \in \mathbb{A}, a \neq b\}$ but $D(|alb) = \{ab \mid a, b \in \mathbb{A}\}$.

► **Remark 3.4.** In fact, the operator N is injective on closed bar languages, because \mathbf{ub} is injective on closed clean bar strings [37, 38]. This means that bar language semantics and global freshness semantics are essentially the same, while local freshness semantics is a quotient of the other semantics. It is immediate from [37, Lemma A.4] that N preserves intersection and complement of closed bar languages, the latter in the sense that $N(\mathbf{bs}(\emptyset) \setminus L) = \mathbb{A}^* \setminus N(L)$ for closed bar languages L . Both properties fail for the local freshness interpretation D ; the semantics of formulae should therefore be understood first in terms of bar languages, with D subsequently applied globally.

4 Syntax and Semantics of Bar- μ TL

We proceed to introduce a variant Bar- μ TL of linear temporal logic whose formulae define bar languages. This logic relates, via its local freshness semantics, to Freeze LTL. It replaces freeze quantification with name binding modalities, and features fixpoints, for increased expressiveness in comparison to the temporal connectives of LTL [17]. Via global freshness semantics, Bar- μ TL may moreover be seen as a logic for session automata [3].

Syntax. We fix a countably infinite set \mathbb{V} of (*fixpoint*) *variables*. The set **Bar** of *bar formulae* ϕ, ψ, \dots (in negation normal form) is generated by the grammar

$$\phi, \psi := \epsilon \mid \neg\epsilon \mid \phi \wedge \psi \mid \phi \vee \psi \mid \heartsuit_\sigma \phi \mid X \mid \mu X. \phi,$$

where $\heartsuit \in \{\diamond, \square\}$, $\sigma \in \bar{\mathbb{A}}$ and $X \in \mathbb{V}$. We define $\top = \epsilon \vee \neg\epsilon$ and $\perp = \epsilon \wedge \neg\epsilon$. We refer to \diamond_σ and \square_σ as σ -*modalities*. The meaning of the Boolean operators is standard; the fixpoint construct μ denotes unique fixpoints, with uniqueness guaranteed by a guardedness restriction to be made precise in a moment. The other constructs are informally described as follows. The constant ϵ states that the input word is empty, and $\neg\epsilon$ that the input word is nonempty. A formula $\diamond_a \phi$ is read “the first letter is a , and the remaining word satisfies ϕ ”, and $\square_a \phi$ is read dually as “if the first letter is a , then the remaining word satisfies ϕ ”. The reading of $|a$ -modalities is similar but involves α -renaming as detailed later in this section; as indicated in Section 3, this means that $|a$ -modalities effectively read fresh letters. They thus replace the freeze quantifier; one important difference with the latter is that $\diamond_{|a}$ consumes the letter it reads, i.e. advances by one step in the input word. A name a is *free* in a formula ϕ if ϕ contains an a -modality at a position that is not in the scope of any $|a$ -modality; that is, $|a$ -modalities bind the name a . We write $\mathbf{FN}(\phi)$ for the set of free names in ϕ , and $\mathbf{BN}(\phi)$ for the set of *bound names* in ϕ , i.e. those names a such that ϕ

mentions $!a$; we put $N(\phi) = \text{FN}(\phi) \cup \text{BN}(\phi)$, and (slightly generously) define the *degree* of ϕ to be $\text{deg}(\phi) = |N(\phi)|$. We write $\text{FV}(\phi)$ for the set of *free* fixpoint variables in ϕ , defined in the standard way by letting μX bind X ; a formula ϕ is *closed* if $\text{FV}(\phi) = \emptyset$. (We refrain from introducing terminology for formulae without free *names*.) As indicated above we require that all fixpoints $\mu X. \phi$ are *guarded*, that is, all free occurrences of X lie within the scope of some σ -modality in ϕ . We denote by $\text{cl}(\phi)$ the *closure* of ϕ in the standard sense [27], i.e. the least set of formulae that contains ϕ and is closed under taking immediate subformulae and unfolding top-level fixpoints; this set is finite. We define the *size* of ϕ as $|\phi| = |\text{cl}(\phi)|$.

For purposes of making Bar a nominal set, we regard every fixpoint variable X with enclosing fixpoint expression $\mu X. \phi$ as being annotated with the set $A = \text{FN}(\mu X. \phi)$; that is, we identify X with the pair (X, A) . We then let G act by replacing names in the obvious way; i.e. $\pi \cdot \phi$ is obtained from ϕ by replacing a with $\pi(a)$, $!a$ with $! \pi(a)$, and (X, A) with $(X, \pi \cdot A)$ everywhere. Otherwise, the definition is as expected:

► **Definition 4.1.** α -Equivalence \equiv_α on formulae is the congruence relation generated by

$$\diamond_{!a}\phi \equiv_\alpha \diamond_{!b}\psi \text{ and } \square_{!a}\phi \equiv_\alpha \square_{!b}\psi \quad \text{whenever } \langle a \rangle \phi = \langle b \rangle \psi$$

(cf. Definition 2.1).

► **Remark 4.2.** The point of implicitly annotating fixpoint variables with the free names of the enclosing μ -expression is to block unsound α -renamings: It ensures that, e.g., $\diamond_{!a}(\mu X. (\diamond_{!a} \epsilon \vee \diamond_{!b} X))$ is *not* α -equivalent to $\diamond_{!a}(\mu X. (\diamond_{!a} \epsilon \vee \diamond_{!a} X))$ (as X is actually $(X, \{a\})$), and is required to ensure stability of α -equivalence under fixpoint expansion, recorded next. We note that fixpoint expansion does *not* avoid capture of names; e.g. the expansion of $\mu X. (\diamond_{!a} \epsilon \vee \diamond_{!a} X)$ is $\diamond_{!a} \epsilon \vee \diamond_{!a}(\mu X. (\diamond_{!a} \epsilon \vee \diamond_{!a} X))$.

► **Lemma 4.3.** Let $\mu X. \phi \equiv_\alpha \mu X. \phi'$. Then $\phi[\mu X. \phi/X] \equiv_\alpha \phi'[\mu X. \phi'/X]$.

Proof. Immediate from the fact that by the convention that fixpoint variables are annotated with the free names of their defining formulae, X , $\mu X. \phi$, and $\mu X. \phi'$ have the same free names and hence allow the same α -renamings in the outer contexts ϕ and ϕ' , respectively. ◀

Semantics. We interpret each bar formula ϕ as denoting a uniformly finitely supported bar language, depending on a *context*, i.e. a finite set $S \subseteq \mathbb{A}$ such that $\text{FN}(\phi) \subseteq S$, which specifies names that are allowed to occur freely; at the outermost level, S will be empty (cf. Section 3). The context grows when we traverse modalities $\diamond_{!a}$ or $\square_{!a}$. Correspondingly, we define satisfaction $S, w \models \phi$ of a formula ϕ by a bar string $w \in \text{bs}(S)$ recursively by the usual clauses for the Boolean connectives, and

$$\begin{aligned} S, w \models \neg \epsilon &\Leftrightarrow w \neq \epsilon \\ S, w \models \epsilon &\Leftrightarrow w = \epsilon \\ S, w \models \mu X. \phi &\Leftrightarrow S, w \models \phi[\mu X. \phi/X] \\ S, w \models \diamond_a \phi &\Leftrightarrow \exists v. w = av \text{ and } S, v \models \phi \\ S, w \models \square_a \phi &\Leftrightarrow \forall v. \text{if } w = av \text{ then } S, v \models \phi \\ S, w \models \diamond_{!a} \phi &\Leftrightarrow \exists \psi \in \text{Bar}, v \in \overline{\mathbb{A}}^*, b \in \mathbb{A}. \\ &\quad w \equiv_\alpha !bv \text{ and } \langle a \rangle \phi = \langle b \rangle \psi \text{ and } S \cup \{b\}, v \models \psi \\ S, w \models \square_{!a} \phi &\Leftrightarrow \forall \psi \in \text{Bar}, v \in \overline{\mathbb{A}}^*, b \in \mathbb{A}. \\ &\quad \text{if } w \equiv_\alpha !bv \text{ and } \langle a \rangle \phi = \langle b \rangle \psi \text{ then } S \cup \{b\}, v \models \psi. \end{aligned}$$

Guardedness of fixpoint variables guarantees that on the right hand side of the fixpoint clause, $\mu X. \phi$ is evaluated only on words that are strictly shorter than w , so the given clause uniquely defines the semantics. Notice that $\diamond_{|a}$ and $\square_{|a}$ allow α -renaming of both the input word and the formula; we comment on this point in Remark 4.7. For a formula ϕ such that $\text{FN}(\phi) = \emptyset$, we briefly write

$$\llbracket \phi \rrbracket_0 = \{w \in \text{bs}(\emptyset) \mid \emptyset, w \models \phi\} \quad \text{and} \quad \llbracket \phi \rrbracket = \llbracket \phi \rrbracket_0 / \equiv_\alpha,$$

referring to $\llbracket \phi \rrbracket_0$ as the *literal language* and to $\llbracket \phi \rrbracket$ as the *bar language* of ϕ (variants with non-empty context and $\text{FN}(\phi) \neq \emptyset$ are technically unproblematic but require more notation). In particular, $\llbracket \phi \rrbracket$ is closed by construction. The *global* and *local freshness semantics* of ϕ are $N(\llbracket \phi \rrbracket)$ and $D(\llbracket \phi \rrbracket)$, respectively, where N and D are the operations converting bar languages into data languages described in Section 3.

► **Remark 4.4.** In $\text{Bar-}\mu\text{TL}$, fixpoints take on the role played by the temporal operators in Freeze LTL. In bar language semantics, the overall mode of expression in $\text{Bar-}\mu\text{TL}$, illustrated in Example 4.8, is slightly different from that of Freeze LTL, as in $\text{Bar-}\mu\text{TL}$ the input is traversed using modalities tied to specific letters rather than using a *next* operator \circ . In local freshness semantics, the effect of \circ is included in the name binding modality $\diamond_{|a}$. For instance, in local freshness semantics we can express LTL-style formulae $\phi U \psi$ (ϕ until ψ) as $\mu X. \psi \vee (\phi \wedge \diamond_{|a} X)$. In particular, $\mu X. \epsilon \vee \diamond_{|a} X$ defines the universal data language, so \top is not actually needed in local freshness semantics. Overall, Freeze LTL and $\text{Bar-}\mu\text{TL}$ (with local freshness semantics) intersect as indicated but are incomparable: On the one hand, Freeze LTL can express the language “the first two letters are different”, which as indicated in Section 3 is not induced by a bar language. On the other hand, $\text{Bar-}\mu\text{TL}$ features fixpoints, which capture properties that generally fail to be expressible using LTL operators, e.g. the language of all even-length words. The latter point relates to the fact that even over finite alphabets, LTL on finite words is only as expressive as first-order logic, equivalently star-free regular expressions (cf. [17]). Constrastingly, thanks to the fixpoint operators, $\text{Bar-}\mu\text{TL}$ is as expressive as its corresponding automata model (Theorem 6.6).

► **Remark 4.5.** As indicated previously, $\text{Bar-}\mu\text{TL}$ is closed under complement: By taking negation normal forms, we can define $\neg\phi$ so that $S, w \models \neg\phi$ iff $S, w \not\models \phi$.

We note next that literal languages of formulae are closed under α -equivalence, and that α -equivalent renaming of formulae indeed does not affect the semantics (cf. Remark 4.2):

► **Lemma 4.6.** For $\phi, \psi \in \text{Bar}$, $a \in \mathbb{A}$, $S \subseteq \mathbb{A}$, and $w, w' \in \overline{\mathbb{A}}^*$, we have:

1. If $S, w \models \psi$ and $w \equiv_\alpha w'$, then $S, w' \models \psi$.
2. If $S, w \models \psi$ and $\phi \equiv_\alpha \phi'$, then $S, w' \models \phi'$.

The proof is by induction along the recursive definition of the semantics; the case for fixpoints in Claim 2 is by Lemma 4.3.

► **Remark 4.7.** We have noted above that the semantics allows α -renaming of both words and formulae. Let us refer to an alternative semantics where the definition of $S, w \models \diamond_{|a} \phi$ is modified to require that there exists $w \equiv_\alpha lav$ such that $S \cup \{a\}, v \models \phi$ (without allowing α -renaming of $\diamond_{|a} \phi$), similarly for $\square_{|a}$, as the *rigid* semantics, and to the semantics defined above as the *actual* semantics. The rigid semantics is not equivalent to the actual semantics, and has several flaws. First off, claim 2 of the above Lemma 4.6 fails under the rigid semantics, in which, for example,

$$\emptyset, l|b|a \models \diamond_{|b} \diamond_{|a} \top \quad \text{but} \quad \emptyset, l|b|a \not\models \diamond_{|b} \diamond_{|b} \top$$

(the latter because $\{b\}, lab \not\models \diamond_{lb}\top$, as α -renaming of la into lb is blocked in lab). More importantly, the rigid semantics has undesirable effects in connection with fixpoints. For instance, in the actual semantics, the formula $\phi = \mu X. ((\neg\epsilon \wedge \square_{la}\perp) \vee \diamond_{la}X)$ has the intuitively intended meaning: A bar string satisfies ϕ iff it contains some plain name. In the rigid semantics, however, we unexpectedly have $\emptyset, labab \not\models \phi$; to see this, note that $\{a\}, labab \not\models \phi$ in the rigid semantics, since α -renaming of lb into la is blocked in $labab$.

► **Example 4.8.** We consider some Bar- μ TL formulae and their respective semantics under local and global freshness. (The local freshness versions are expressible in Freeze LTL in each case; recall however Remark 4.4.)

1. The bar language $\llbracket \top \rrbracket$ is the set of all closed bar strings (modulo α -equivalence, a qualification that we omit henceforth). Under both global and local freshness semantics, this becomes the set of all data words.
2. The bar language $\llbracket \diamond_{la}\square_a\epsilon \rrbracket$ is the language of all closed bar strings that start with a bar name la , and stop after the second letter if that letter exists and is the plain name a (e.g. $\llbracket \diamond_{la}\square_a\epsilon \rrbracket$ contains $la, laa, labab$ but not $laaa$). In both global and local freshness semantics, this becomes the language of all words that stop after the second letter if that letter exists and coincides with the first letter.
3. In context $\{a\}$, a bar string satisfies $\mu Y. ((\diamond_{lb}Y) \vee \diamond_a\top)$ iff it contains a free occurrence of a preceded only by bar names distinct from la . Thus, the bar language of

$$\mu X. (\diamond_{la}(X \vee \mu Y. ((\diamond_{lb}Y) \vee \diamond_a\top)))$$

consists of all closed bar strings that start with a prefix of bar names and eventually mention a plain name corresponding to one of these bar names. Under both local and global freshness semantics, this becomes the data language of all words mentioning some letter twice (which is not acceptable by deterministic or even unambiguous register automata [2, 37]). Notice that during the evaluation of the formula, the context can become unboundedly large, as it grows every time a bar name is read.

4. The bar language of the similar formula

$$\mu X. ((\diamond_{la}X) \vee (\diamond_{la}\mu Y. ((\diamond_{lb}Y) \vee \diamond_a\epsilon)))$$

consists of all closed bar strings where all names except the last one are bound names. Under global freshness semantics, this becomes the data language where the last letter occurs *precisely* twice in the word, and all other names only once. Under local freshness semantics, the induced data language is that of all words where the last letter occurs *at least* twice, with no restrictions on the other letters.

5. To illustrate both the mechanism of local freshness via α -equivalence and, once again, the use of \top , we consider the bar language of

$$\diamond_{la}\diamond_{lb}\mu X. ((\diamond_{lb}X) \vee \diamond_a\top),$$

which consists of all closed bar strings that start with a bar name la , at some later point contain a substring bab , and have only bar names distinct from la in between. Under global freshness semantics, this becomes the data language of all words where the first name a occurs a second time at the third position or later, all letters are mutually distinct until that second occurrence, and the letter preceding that occurrence is repeated immediately after. The local freshness semantics is similar but only requires the letters between the first and second occurrence of a to be distinct from a (rather than mutually distinct), that is, the substring bab is required to contain precisely the second occurrence of a .

5 Extended Regular Nondeterministic Nominal Automata

We proceed to introduce the nominal automaton model we use in model checking, *extended regular nondeterministic nominal automata* (ERNNAs), a generalized version of RNNAs [38] that allow for limited alternation in the form of deadlocked universal states.

Nominal automata models [2] generally feature nominal sets of states; these are infinite as sets but typically required to be orbit-finite. RNNAs are distinguished from other nominal automata models (such as *nondeterministic orbit-finite automata* [2]) in that they impose finite branching but feature *name-binding* transitions; that is, they have *free* transitions $q \xrightarrow{a} q'$ for $a \in \mathbb{A}$ as well as *bound* transitions $q \xrightarrow{la} q'$, both consuming the respective type of letter in the input bar string w . Bound transitions may be understood as reading fresh letters. RNNAs are a nondeterministic model, i.e. accept w if there *exists* a run on w ending in an accepting state. ERNNAs additionally feature \top -states that accept the current word even if it has not been read completely, and thus behave like the formula \top ; these states may be seen as universal states without outgoing transitions. Formal definitions are as follows.

► **Definition 5.1.** An *extended regular nondeterministic nominal automaton* (ERNNA) is a four-tuple $A = (Q, \rightarrow, s, f)$ that consists of

- an orbit-finite nominal set Q of *states* (whose orbits we also refer to as the *orbits of A*);
- an *initial state* $s \in Q$ such that $\text{supp}(s) = \emptyset$;
- an equivariant *transition relation* $\rightarrow \subseteq Q \times (\overline{\mathbb{A}} \cup \{\epsilon\}) \times Q$, with $(q, \sigma, q') \in \rightarrow$ denoted by $q \xrightarrow{\sigma} q'$; and
- an equivariant *acceptance function* $f: Q \rightarrow \{0, 1, \top\}$

such that \rightarrow is α -invariant (that is, $q \xrightarrow{la} q'$ and $\langle a \rangle q' = \langle b \rangle q''$ imply $q \xrightarrow{lb} q''$) and finitely branching up to α -equivalence (i.e. for each q , the sets $\{(a, q') \mid q \xrightarrow{a} q'\}$, $\{(\epsilon, q') \mid q \xrightarrow{\epsilon} q'\}$, and $\{\langle a \rangle q' \mid q \xrightarrow{la} q'\}$ are finite). Whenever $f(q) = \top$, we require $\text{supp}(q) = \emptyset$ and moreover that q is a deadlock, i.e. there are no transitions of the form $q \xrightarrow{\sigma} q'$. The *degree* $\text{deg}(A)$ of A is the maximal size of the support of a state in Q (in the translation of nominal automata into register automata, the degree corresponds to the number of registers [2, 38]). A state q is *accepting* if $f(q) = 1$, *non-accepting* if $f(q) = 0$, and a \top -state if $f(q) = \top$.

We extend the transition relation to words w over $\overline{\mathbb{A}}$, i.e. to bar strings, as usual; that is, $q \xrightarrow{w} q'$ iff there exist states $q = q_0, q_1, \dots, q_k = q'$ and transitions $q_i \xrightarrow{\sigma_{i+1}} q_{i+1}$ for $i = 0, \dots, k-1$ such that w is the concatenation $\sigma_1 \cdots \sigma_k$, where σ_i is regarded as a one-letter word if $\sigma_i \in \overline{\mathbb{A}}$, and as the empty word if $\sigma_i = \epsilon$. We define $L_{\text{pre}}(A) \subseteq \overline{\mathbb{A}}^* \times \{1, \top\}$ (for *prelanguage*) as

$$L_{\text{pre}}(A) = \{(w, f(q)) \mid s \xrightarrow{w} q, f(q) \in \{1, \top\}\}.$$

The *literal language* $L_0(A) \subseteq \overline{\mathbb{A}}^*$ accepted by an ERNNA A then is defined by

$$L_0(A) = \text{bs}(\emptyset) \cap (\{w \mid (w, 1) \in L_{\text{pre}}(A)\} \cup \{vu \mid (v, \top) \in L_{\text{pre}}(A), u \in \overline{\mathbb{A}}^*\});$$

that is, a closed bar string w is literally accepted if either w has a run ending in an accepting state or a prefix of w has a run ending in a \top -state. (Again, extending the treatment to bar strings with free names is technically unproblematic but heavier on notation.) The *bar language accepted by A* is the quotient

$$L_\alpha(A) = L_0(A) / \equiv_\alpha.$$

We say that A is ϵ -free if A contains no ϵ -transitions. If A is ϵ -free and contains no \top -states, then A is a *regular nondeterministic nominal automaton* (RNNA).

► **Remark 5.2.** The presence of \top -states makes ERNNAs strictly more expressive than RNNAs under bar language semantics (equivalently, under global freshness semantics). Indeed, the ERNNA consisting of a single \top -state accepts the universal bar language, which is not acceptable by an RNNNA [38]. On the other hand, under local freshness semantics, an accepting state with a $!a$ -self-loop accepts the universal data language (in analogy to the expressibility of \top by $\mu X. \epsilon \vee \diamond_{!a} X$ in $\text{Bar-}\mu\text{TL}$ under local freshness semantics, cf. Remark 4.4), so RNNAs are as expressive as ERNNAs under local freshness semantics.

Name dropping. Like for RNNAs, the literal language accepted by an ERNNA is not in general closed under α -equivalence. However, one can adapt the notion of name dropping [38] to ERNNA: Roughly speaking, an ERNNA is *name-dropping* if all its transitions may nondeterministically lose any number of names from the support of states (which corresponds to losing register contents in a register automaton). The literal language of a name-dropping ERNNA is closed under α -equivalence, and every ERNNA A can be transformed into a name-dropping ERNNA $\text{nd}(A)$, preserving the bar language. This transformation is central to the inclusion checking algorithm (see additional remarks in Section 7).

Representing ERNNAs. ERNNAs are, *prima facie*, infinite objects; we next discuss a finite representation of ERNNAs as *extended bar NFAs*, generalizing the representation of RNNAs as bar NFAs [38]. The intuition behind extended bar NFAs is similar to that of ERNNAs, except that extended bar NFAs are not closed under name permutation. In particular, extended bar NFAs feature deadlocked universal states:

► **Definition 5.3.** An *extended bar NFA* $A = (Q, \rightarrow, s, f)$ consists of

- a finite set Q of *states*;
- a *transition relation* $\rightarrow \subseteq (Q \times \bar{\mathbb{A}} \times Q)$, with $(q, \sigma, q') \in \rightarrow$ denoted by $q \xrightarrow{\sigma} q'$;
- an *initial state* $s \in Q$; and
- an *acceptance function* $f: Q \rightarrow \{0, 1, \top\}$

such that whenever $f(q) = \top$, then q is a deadlock, i.e. has no outgoing transitions. We extend the transition relation to words over $\bar{\mathbb{A}}$ (including the empty word) as usual. Similarly as for ERNNAs, we define $L_{\text{pre}}(A) \subseteq \bar{\mathbb{A}}^* \times \{1, \top\}$ by

$$L_{\text{pre}}(A) = \{(w, f(q)) \mid s \xrightarrow{w} q, f(q) \in \{1, \top\}\}.$$

The *literal language* $L_0(A)$ accepted by A is

$$L_0(A) = \{w \mid (w, 1) \in L_{\text{pre}}(A)\} \cup \{vu \mid (v, \top) \in L_{\text{pre}}(A), u \in \bar{\mathbb{A}}^*\}.$$

The *bar language* of A is then defined as the quotient $L_\alpha(A) = L_0(A)/\equiv_\alpha$. For ease of presentation, we only consider the case where $L_\alpha(A)$ is closed, which is easily checked syntactically (no a may be reached in A without passing $!a$). We generally write (A, q) for the extended bar NFA that arises by making $q \in Q$ the initial state of A , dropping however the requirement that the bar language accepted by (A, q) is closed. The set $\text{FN}(A, q)$ of *free names* of $q \in Q$ is then $\text{FN}(A, q) = \bigcup_{(w,b) \in L_{\text{pre}}(A,q)} \text{FN}(w)$. Slightly sharpening the original definition [38], we take the *degree* of A to be $\text{deg}(A) := \max_{q \in Q} |\text{FN}(A, q)|$.

► **Theorem 5.4.** *ERNNAs and extended bar NFAs accept the same bar languages; that is:*

1. *For a given extended bar NFA with n states and degree k , there exists a name-dropping ERNNA of degree k with $n \cdot 2^k$ orbits that accepts the same bar language.*
2. *For a given ERNNA with n orbits and degree k , there exists an extended bar NFA of degree k with $n \cdot k!$ states that accepts the same bar language.*

The key algorithmic task on ERNNAs is inclusion checking; we generalize the inclusion algorithm for RNNAs [38] to obtain

► **Theorem 5.5.** *Given a bar NFA A_1 and an extended bar NFA A_2 , the inclusion $L_\alpha(A_1) \subseteq L_\alpha(A_2)$ can be checked using space polynomial in the number of orbits of A_1 and A_2 , and exponential in $\deg(A_1)$ and $\deg(A_2)$. The same holds under local freshness semantics.*

6 Equivalence of Bar- μ TL and ERNNA

Our model checking algorithm will be based on translation of closed formulae into ERNNAs, in what amounts to a tableau construction that follows a similar spirit as the standard automata-theoretic translation of LTL, but requires a special treatment of \Box -formulae and $\neg\epsilon$, and moreover uses nondeterminism to bound the number of free names in automata states (which may be thought of as the number of registers) by guessing certain names, as explained in the following example.

► **Example 6.1.** Consider the formulae $\phi(b) = \mu Y. (\Box_b \perp \wedge \Box_{!c} Y)$ and $\psi = \mu X. (\Box_{!a} X \wedge \Box_{!b} \phi(b))$. The formula $\phi(b)$ states that the first plain name that occurs is not a free occurrence of b , and ψ thus states that none of the bar names have a free occurrence later on. When evaluating ψ over a bar string $w = |a_1 a_2 \dots |a_n a_i v$ consisting of n bar names $|a_i$ followed by the plain name a_i ($1 \leq i \leq n$) and a remaining bar string v (so w does not satisfy ψ), one eventually has to evaluate all the formulae $\phi(a_1), \dots, \phi(a_n)$ over the bar string $a_i v$. Thus, the number of copies of formulae that a naively constructed ERRNA for ψ needs to keep track of can in principle grow indefinitely. At a first glance, this seems to prohibit a translation of formulae into orbit-finite automata. However, we observe that when the letter a_i is read, all $\phi(a_j)$ for $i \neq j$ immediately evaluate to \top (since the conjuncts $\Box_{a_j} \perp$ and $\Box_{!c} \phi(a_j)$ of their fixpoint unfolding both hold vacuously), and only the evaluation of $\phi(a_i)$ becomes relevant (since the argument of $\Box_{a_i} \perp$ is actually evaluated). In fact, it is possible to let the ERRNA for ψ nondeterministically guess the first plain name a_i that occurs in the input bar string. Then it suffices to let the ERNNA keep track of $\phi(a_i)$ since as discussed, all other copies of $\phi(b)$ become irrelevant.

The idea from the previous example can be generalized to work for all formulae. To this end we introduce a recursive manipulation of formulae that uses annotations to explicitly restrict the support of formulae and to guess and enforce so-called distinguishing letters. When constructing an ERNNA from a formula, we will use such manipulated formulae to avoid the problem described in Example 6.1 by bounding the number of formulae that the constructed ERNNA has to track.

► **Definition 6.2.** Fix a marker element $* \notin \mathbb{A}$ (indicating absence of a name). Let ϕ be a formula, let B and C be sets of letters such that $B \subseteq C$, and let $a \in (C \setminus B) \cup \{*\}$. For $n \in \mathbb{N}$, we define $\phi_C^B(a)_n$ recursively (as termination measure of the recursive definition we use tuples $(|n|, \mathbf{u}(\phi), |\phi|)$, ordered by lexicographic ordering, where $\mathbf{u}(\phi)$ denotes the number of unguarded fixpoint operators in ϕ) by putting $\phi_C^B(a)_0 = \top$ and, for $n > 0$,

$$\begin{aligned} \epsilon_C^B(a)_n &= \epsilon & \neg\epsilon_C^B(a)_n &= \neg\epsilon \\ (\psi \wedge \chi)_C^B(a)_n &= \psi_C^B(a)_n \wedge \chi_C^B(a)_n & (\psi \vee \chi)_C^B(a)_n &= \psi_C^B(a)_n \vee \chi_C^B(a)_n \\ (\diamond_{!b} \psi)_C^B(a)_n &= \diamond_{!b} (\psi_{C \cup \{b\}}^{B \cup \{b\}}(a)_{n-1}) & (\Box_{!b} \psi)_C^B(a)_n &= \Box_{!b} (\psi_{C \cup \{b\}}^{B \cup \{b\}}(a)_{n-1}) \\ (\mu X. \psi)_C^B(a)_n &= (\psi[\mu X. \psi/X])_C^B(a)_n \end{aligned}$$

and

$$\begin{aligned}
 (\diamond_b \psi)_C^B(a)_n &= \begin{cases} \perp & b \notin C \\ \diamond_b(\psi_C^B(a)_{n-1}) & b \in C, b \in B \\ \diamond_b(\psi_{\text{FN}(\psi)}^\emptyset(*)_{n-1}) & b \in C, b \notin B \end{cases} \\
 (\square_b \psi)_C^B(a)_n &= \begin{cases} \top & b \notin C \\ \square_b(\psi_C^B(a)_{n-1}) & b \in C, b \in B \\ \chi(b)_C^B(a)_{n-1} & b \in C, b \notin B \end{cases}
 \end{aligned}$$

where

$$\chi(b)_C^B(a)_{n-1} = \begin{cases} \square_b(\psi_{\text{FN}(\psi)}^\emptyset(*)_{n-1}) & a = * \\ \epsilon \vee \diamond_{lc} \top \vee \bigvee_{d \in B} \diamond_d \top \vee \diamond_a(\psi_{\text{FN}(\psi)}^\emptyset(*)_{n-1}) & a = b \\ \epsilon \vee \diamond_{lc} \top \vee \bigvee_{d \in B \cup \{a\}} \diamond_d \top & * \neq a \neq b. \end{cases}$$

During this process, fixpoint formulae are unfolded before replacing any free modalities within their arguments; by guardedness of fixpoint variables, this happens at most n times. (We intend the number n as a strict upper bound on the length of bar strings over which $\phi_C^B(a)_n$ is meant to be evaluated; cf. Lemma 6.3.) The process replaces just the first freely occurring boxes whose index is in C but not in B ; hence, modal operators whose index comes from B are left unchanged. Intuitively, $\phi_C^B(a)_n$ is a formula that behaves like the restriction of ϕ to the support C on bar strings w such that $|w| < n$ and in which a is the first free name that is not contained in B (if any such name occurs in w ; in this case we refer to the letter a as *distinguishing letter*). Hence $\phi_C^B(a)_n$ is like the restriction of ϕ to support C , assuming that the distinguishing letter is a . Formally:

► **Lemma 6.3.** *Let B and C be sets of names such that $B \subseteq C$, let $a \in (C \setminus B) \cup \{*\}$, let ϕ be a formula, and let S be a context. Let v be a bar string such that if $a \neq *$, then each letter that has a free occurrence in v before the first free occurrence of a in v is contained in B . Also, suppose that if there is some freely occurring letter in v that is not contained in B and the first such letter d is in $\text{FN}(\phi)$, then $d \in C$. Under these assumptions, we have*

$$S, v \models \phi \iff S, v \models \phi_C^B(a)_n \quad \text{for all } n \text{ such that } |v| < n.$$

Proof sketch. Induction along the recursive definition of $\phi_C^B(a)_n$. ◀

Let ϕ be a formula, let B , C , and D be sets of names such that $B \subseteq C$, and let $a \in (C \setminus B) \cup D \cup \{*\}$. We put

$$\phi_C^B(a)_n^D = \begin{cases} \perp & \text{if } a \in D \\ \phi_C^B(a)_n & \text{if } a \notin D, \end{cases}$$

the intuition being that $\phi_C^B(a)_n^D$ encodes ϕ (restricted to support C) together with the guess that a is the distinguishing letter (and that all names freely occurring before a are from the set B), where guessing a letter from the set D is not allowed. This rules out the situation that a letter from D is used to satisfy a distinguishing box formula in ϕ . Using Lemma 6.3, we obtain:

► **Lemma 6.4.** *Let π, π' be permutations, let ϕ be a formula, let w be a bar string, let S be a context, and put $A := \text{FN}(\pi \cdot \phi)$, $A' := \text{FN}(\pi' \cdot \phi)$ and $B := A \cap A'$. Furthermore, let a be either the first freely occurring letter in w that is not in B if that letter exists and is in $A \cup A'$ (in which case we say that a is the distinguishing letter w.r.t B and w), and $a = *$ otherwise. Then $a \notin B$ and we have*

$$S, w \models (\pi \cdot \phi) \wedge (\pi' \cdot \phi) \iff S, w \models ((\pi \cdot \phi)_A^B(a)_{|w|+1}^{A'} \wedge (\pi' \cdot \phi)_B^\emptyset(*)_{|w|+1}^\emptyset) \vee ((\pi' \cdot \phi)_{A'}^B(a)_{|w|+1}^A \wedge (\pi \cdot \phi)_B^\emptyset(*)_{|w|+1}^\emptyset).$$

Relying on name restriction as in Lemma 6.4, we are able to translate formulae to ERRNAs.

► **Theorem 6.5.** *For every closed formula ϕ of size m and degree k , there is an ERNNA $A(\phi)$ with degree bounded exponentially in k and polynomially in m , and with number of orbits bounded doubly exponentially in k and singly exponentially in m , that accepts the bar language of ϕ , i.e. $L_\alpha(A(\phi)) = \llbracket \phi \rrbracket$.*

We sketch the construction of $A(\phi)$. Let $\text{cl}(\phi)$ denote the closure of ϕ , defined in the standard way. We put

$$\text{Cl} = \text{cl}(\phi) \cup \{\diamond_\sigma \psi \mid \square_\sigma \psi \in \text{cl}(\phi)\} \cup \{\epsilon, \perp\} \cup \{\diamond_b \top \mid b \in \text{BN}(\phi)\},$$

noting $|\text{Cl}| \leq 4m$ (recall that \perp and \top abbreviate $\epsilon \wedge \neg \epsilon$ and $\epsilon \vee \neg \epsilon$, respectively). Furthermore, we put

$$\text{formulae} = \{\psi_C^B(a) \mid \psi \in \text{Cl}, B \subseteq \text{N}(\phi), C \subseteq \text{FN}(\psi), a \in C \cup \{*\}\},$$

noting that the cardinality of formulae is linear in m and exponential in k ; specifically, $|\text{formulae}| \leq |\text{Cl}| \cdot (|\text{N}(\phi)| + 1) \cdot 2^{2|\text{N}(\phi)|} \leq 2^{2k} \cdot 4m(k+1) \leq 2^{2m} \cdot 5m^2$ since we have $|\text{FN}(\psi)| \leq |\text{N}(\phi)|$ for all $\psi \in \text{Cl}$ and since $k \leq m$. The set formulae contains formulae $\psi \in \text{Cl}$ that are annotated with a single name a (or $*$) and two sets B and C of names. The annotation with a is used to encode a guessed name (with $*$ denoting the situation that no name has been guessed yet) which we call *distinguishing letter*, while the set C encodes the restriction of the support of ψ to C and the set B denotes the names that are allowed to occur freely before the distinguishing letter does. This data will be used to bound the number of copies of subformulae that can occur in nodes of the constructed tableau. We construct an ERRNA $A(\phi) = (Q, \rightarrow, s, f)$ with carrier set

$$Q = \left\{ \left(\{(\pi_1 \cdot \phi_1, \dots, \pi_n \cdot \phi_n), a\} \mid \pi_1, \dots, \pi_n \in G, \right. \right. \\ \left. \left. \{\phi_1, \dots, \phi_n\} \subseteq \text{formulae}, a \in \mathbb{A} \cup \{*\} \right\},$$

where the π_i act on names as usual and we define $\pi_i(*) = *$. The bound on the number of orbits follows since each combination of a subset Φ of formulae, a name a and an equivalence relation on the set of free names of Φ and a gives rise to at most one orbit. We put $s = (\{\phi_{\text{FN}(\phi)}^\emptyset(*)\}, *)$. Given a state $(\Gamma, b) \in Q$, formulae $\pi \cdot (\psi_C^B(a)) \in \Gamma$ stand for instances of ψ in which the distinguishing letter is fixed to be $\pi \cdot a$, $\pi \cdot B$ is the set of free names that are allowed to occur before $\pi \cdot a$ does, and the support of ψ is restricted to $\pi \cdot C$. The shape of a formula $\pi \cdot (\psi_C^B(a))$ is just the shape of ψ ; for brevity, we omit the annotations with C, B, a when they are not relevant. To deal with box operators and $\neg \epsilon$, states also contain a separate name component b (which may be $*$) that denotes a guess of the last relevant free name after which the evaluation of formulae from Γ will stop. A state $(\Gamma, b) \in Q$ is *propositional* if Γ contains some formula of the shape $\psi_1 \vee \psi_2$, $\psi_1 \wedge \psi_2$, or $\mu X. \psi_1$; *quasimodal* if (Γ, b) is not propositional but Γ contains some formula of the form $\neg \epsilon$ or $\square_\sigma \phi$; and *modal* if (Γ, b) is

neither propositional nor quasimodal, i.e. if all elements of Γ are either of the shape ϵ or $\diamond_\sigma\phi$. We define the acceptance function $f: Q \rightarrow \{0, 1, \top\}$ by $f(\emptyset, b) = \top$, $f(\Gamma, b) = 1$ if $\Gamma \neq \emptyset$ and all elements of Γ are of the shape ϵ , and $f(\Gamma, b) = 0$ for all other states.

Transitions work roughly as follows. The component Γ of a state (Γ, b) plays the usual role: It records formulae that the automaton requires the remaining input word to satisfy. To bound the number of free names that have to be tracked, formulae are annotated with guesses for distinguishing letters. The transitions from propositional and modal states follow the standard tableau rules; e.g. given a propositional state $q = (\Gamma \cup \{\psi \wedge \chi\}, b)$, we have $q \xrightarrow{\epsilon} (\Gamma \cup \{\psi, \chi\}, b)$; given a modal state $q = (\{\diamond_a\psi_1, \dots, \diamond_a\psi_n\}, b)$, we have $q \xrightarrow{a} (\{\psi_1, \dots, \psi_n\}, b)$; and a modal state containing $\diamond_a\psi$ and $\diamond_b\chi$ for $a \neq b$ is a deadlock (the treatment of \diamond_{1a} is more involved). \Box -formulae and $\neg\epsilon$ are dealt with by ϵ -transitions from quasimodal states (Γ, b) to modal states. This process is straightforward if Γ contains some formula $\diamond_\sigma\psi$. The critical case is the remaining one: A formula $\Box_\sigma\psi$ in Γ can be satisfied by in fact satisfying $\diamond_\sigma\psi$, by ending the word, or by reading either a plain name c other than σ or a bar name (if $\sigma \in \mathbb{A}$), or by reading any plain name (if $\sigma \in \overline{\mathbb{A}}$). This is where the second component b of states comes in: The letter c must have previously appeared as a bar name; $A(\phi)$ guesses when this happens (in bound transitions from modal states), and records its guess in b , with $*$ representing the situation that no guess has yet been made.

When constructing transitions in $A(\phi)$ as explained above, the set of formulae to which a naively constructed transition leads may contain two instances $\pi \cdot (\psi_C^B(a))$ and $\pi' \cdot (\psi_C^B(a))$ of an annotated formula ψ . In this situation, the rest of the word has to satisfy both formulae, but the set Q can only contain one instance of $\psi_C^B(a)$. Here we use the above restriction technique and repeated application of Lemma 6.4 to ensure that only a single copy needs to be kept.

We also have a converse translation which goes via the equivalence of ERNNAs and extended bar NFAs, and associates a fixpoint variable to each state.

► **Theorem 6.6.** *For every ERNNA A there exists a formula ϕ such that $\llbracket \phi \rrbracket = L_\alpha(A)$.*

Notably, by Remark 4.5, the results of this section imply

► **Corollary 6.7.** *The class of closed bar languages definable by ERNNAs is closed under complement: for each ERNNA A there exists an ERNNA \bar{A} such that $L_\alpha(\bar{A}) = \text{bs}(\emptyset) \setminus L_\alpha(A)$.*

As mentioned previously, RNNAs are essentially equivalent to session automata [38], which are only closed under *resource-bounded* complement [3] for some resource bound k ; in our present terminology, this corresponds to complement within the set of closed bar strings, modulo α -equivalence, that can be written with at most k different bound names. It is maybe surprising that full complementation (of session automata or RNNAs) is enabled by simply allowing \top -states.

7 Reasoning for Bar- μ TL

Using Theorem 6.5, we reduce reasoning problems for Bar- μ TL to ERNNAs:

► **Definition 7.1** (Reasoning problems). A closed formula ϕ is *satisfiable* if $\llbracket \phi \rrbracket \neq \emptyset$, and *valid* if $\llbracket \phi \rrbracket = \text{bs}(\emptyset)$. A formula ψ *refines* ϕ if $\llbracket \psi \rrbracket \subseteq \llbracket \phi \rrbracket$. An RNNA A *satisfies* ϕ (written $A \models \phi$) if $L_\alpha(A) \subseteq \llbracket \phi \rrbracket$. The *model checking problem* is to decide whether $A \models \phi$. We sometimes emphasize that these problems refer to *bar languages* or, equivalently, *global freshness*. The corresponding problems for *local freshness* arise by applying the D operator (Section 3) to all bar languages involved; e.g. A *satisfies* ϕ *under local freshness* if $D(L_\alpha(A)) \subseteq D(\llbracket \phi \rrbracket)$.

The complexity of $\text{Bar-}\mu\text{TL}$ reasoning problems, obtained using in particular Theorem 6.5 and Theorem 5.5, is summed up as follows.

► **Theorem 7.2.**

1. Model checking $\text{Bar-}\mu\text{TL}$ over $RNNAs$ is in 2EXPSPACE , more precisely in para-EXPSPACE with the degree of the formula as the parameter, under both bar language semantics (equivalently global freshness) and under local freshness. The same holds for validity under local freshness.
2. The satisfiability problem for $\text{Bar-}\mu\text{TL}$ is in EXPSPACE , more precisely in para-PSPACE with the degree of the formula as the parameter, under both bar language semantics / global freshness and under local freshness. The same holds for validity and refinement under bar language semantics / global freshness.

We leave the complexity (and in fact the decidability) of refinement checking under local freshness semantics as an open problem.

8 Conclusions

We have defined a specification logic $\text{Bar-}\mu\text{TL}$ for finite words over infinite alphabets, modelled in the framework of nominal sets, which covers both local and global freshness. $\text{Bar-}\mu\text{TL}$ features freeze quantification in the shape of name-binding modalities, and as such relates to Freeze LTL. It combines comparatively low complexity of the main reasoning problems with reasonable expressiveness, in particular unboundedly many registers, full nondeterminism, and closure under complement. Freshness is based on α -equivalence in nominal words; this entails certain expressive limitations on local freshness, which however seem acceptable in relation to the mentioned good computational properties.

An important issue for future work is the behaviour of $\text{Bar-}\mu\text{TL}$ over infinite words; also, we will investigate whether our methods extend to languages of data trees (e.g. [13]).

References

- 1 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27:1–27:26, 2011. doi:10.1145/1970398.1970403.
- 2 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3), 2014. doi:10.2168/LMCS-10(3:4)2014.
- 3 Benedikt Bollig, Peter Habermehl, Martin Leucker, and Benjamin Monmege. A robust class of data languages and an application to learning. *Log. Meth. Comput. Sci.*, 10(4), 2014. doi:10.2168/LMCS-10(4:19)2014.
- 4 Benedikt Bollig, Karin Quaas, and Arnaud Sangnier. The complexity of flat freeze LTL. *Log. Methods Comput. Sci.*, 15(3), 2019. doi:10.23638/LMCS-15(3:33)2019.
- 5 Thomas Colcombet. Unambiguity in automata theory. In *Descriptive Complexity of Formal Systems, DCFS 2015*, volume 9118 of *LNCS*, pages 3–18. Springer, 2015. doi:10.1007/978-3-319-19225-3.
- 6 Thomas Colcombet and Amaldev Manuel. Generalized data automata and fixpoint logic. In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2014*, volume 29 of *LIPICs*, pages 267–278. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPICs.FSTTCS.2014.267.
- 7 Thomas Colcombet and Amaldev Manuel. μ -calculus on data words. *CoRR*, 2014. arXiv:1404.4827.

- 8 Thomas Colcombet and Amaldev Manuel. Fragments of fixpoint logic on data words. In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2015*, volume 45 of *LIPICs*, pages 98–111. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.FSTTCS.2015.98.
- 9 Wojciech Czerwiński, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. *J. ACM*, 68(1):7:1–7:28, 2021. doi:10.1145/3422822.
- 10 Wojciech Czerwiński and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. *CoRR*, 2021. arXiv:2104.13866.
- 11 Stéphane Demri and Ranko Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009. doi:10.1145/1507244.1507246.
- 12 Stéphane Demri and Arnaud Sangnier. When model-checking freeze LTL over counter machines becomes decidable. In *Foundations of Software Science and Computation Structures, FOSSACS 2010*, volume 6014 of *LNCS*, pages 176–190. Springer, 2010. doi:10.1007/978-3-642-12032-9_13.
- 13 Diego Figueira and Luc Segoufin. Future-looking logics on data words and trees. In *Mathematical Foundations of Computer Science, MFCS 2009*, volume 5734 of *LNCS*, pages 331–343. Springer, 2009. doi:10.1007/978-3-642-03816-7_29.
- 14 Murdoch J. Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bull. Symb. Log.*, 17(2):161–229, 2011. doi:10.2178/bs1/1305810911.
- 15 Murdoch J. Gabbay and Andrew M. Pitts. A new approach to abstract syntax involving binders. In *Logic in Computer Science, LICS 1999*, pages 214–224. IEEE Computer Society, 1999. doi:10.1109/LICS.1999.782617.
- 16 Murdoch James Gabbay and Vincenzo Ciancia. Freshness and name-restriction in sets of traces with names. In *Foundations of Software Science and Computation Structures, FOSSACS 2011*, volume 6604 of *LNCS*, pages 365–380. Springer, 2011. doi:10.1007/978-3-642-19805-2.
- 17 Giuseppe De Giacomo and Moshe Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *International Joint Conference on Artificial Intelligence, IJCAI 2013*, pages 854–860. IJCAI/AAAI, 2013. URL: <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997>.
- 18 Radu Grigore, Dino Distefano, Rasmus Petersen, and Nikos Tzevelekos. Runtime verification based on register automata. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2013*, volume 7795 of *LNCS*, pages 260–276. Springer, 2013. doi:10.1007/978-3-642-36742-7_19.
- 19 Jan Groote and Radu Mateescu. Verification of temporal properties of processes in a setting with data. In *Algebraic Methodology and Software Technology, AMAST 1998*, volume 1548 of *LNCS*, pages 74–90. Springer, 1998. doi:10.1007/3-540-49253-4_8.
- 20 Jan Groote and Tim Willemse. Model-checking processes with data. *Sci. Comput. Prog.*, 56(3):251–273, 2005. doi:10.1016/j.scico.2004.08.002.
- 21 Orna Grumberg, Orna Kupferman, and Sarai Sheinvald. Model checking systems and specifications with parameterized atomic propositions. In *Automated Technology for Verification and Analysis, ATVA 2012*, volume 7561 of *LNCS*, pages 122–136. Springer, 2012. doi:10.1007/978-3-642-33386-6_11.
- 22 Daniel Hausmann, Stefan Milius, and Lutz Schröder. Harnessing LTL with freeze quantification. *CoRR*, 2020. arXiv:2010.10912.
- 23 Falk Howar, Bengt Jonsson, and Frits Vaandrager. Combining black-box and white-box techniques for learning register automata. In *Computing and Software Science – State of the Art and Perspectives*, volume 10000 of *LNCS*, pages 563–588. Springer, 2019. doi:10.1007/978-3-319-91908-9_26.
- 24 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

- 25 Bartek Klin, Sławomir Lasota, and Szymon Toruńczyk. Nondeterministic and conondeterministic implies deterministic, for data languages. In *Foundations of Software Science and Computation Structures, FOSSACS 2021*, volume 12650 of *LNCS*, pages 365–384. Springer, 2021. doi:10.1007/978-3-030-71995-1_19.
- 26 Bartek Klin and Mateusz Łętyk. Scalar and vectorial μ -calculus with atoms. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:5)2019.
- 27 Dexter Kozen. Results on the propositional μ -calculus. *Theor. Comput. Sci.*, 27:333–354, 1983. doi:10.1016/0304-3975(82)90125-6.
- 28 Dexter Kozen, Konstantinos Mamouras, Daniela Petrisan, and Alexandra Silva. Nominal Kleene coalgebra. In *Automata, Languages, and Programming, ICALP 2015*, volume 9135 of *LNCS*, pages 286–298. Springer, 2015. doi:10.1007/978-3-662-47666-6.
- 29 Dexter Kozen, Konstantinos Mamouras, and Alexandra Silva. Completeness and incompleteness in nominal kleene algebra. In *Relational and Algebraic Methods in Computer Science, RAMiCS 2015*, volume 9348 of *LNCS*, pages 51–66. Springer, 2015. doi:10.1007/978-3-319-24704-5.
- 30 Klaas Kürtz, Ralf Küsters, and Thomas Wilke. Selecting theories and nonce generation for recursive protocols. In *Formal methods in security engineering, FMSE 2007*, pages 61–70. ACM, 2007. doi:10.1145/1314436.1314445.
- 31 Ranko Lazić. Safely freezing LTL. In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2006*, volume 4337 of *LNCS*, pages 381–392. Springer, 2006. doi:10.1007/11944836_35.
- 32 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. *CoRR*, 2021. arXiv:2104.12695.
- 33 Amaldev Manuel, Anca Muscholl, and Gabriele Puppis. Walking on data words. *Theory Comput. Sys.*, 59(2):180–208, 2016. doi:10.1007/s00224-014-9603-3.
- 34 Antoine Mottet and Karin Quaas. The containment problem for unambiguous register automata. In *Theoretical Aspects of Computer Science, STACS 2019*, volume 126 of *LIPICs*, pages 53:1–53:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.STACS.2019.53.
- 35 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004. doi:10.1145/1013560.1013562.
- 36 Andrew Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, 2013.
- 37 Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. Nominal automata with name binding. *CoRR*, 2016. arXiv:1603.01455.
- 38 Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. Nominal automata with name binding. In *Foundations of Software Science and Computation Structures, FOSSACS 2017*, volume 10203 of *LNCS*, pages 124–142, 2017. doi:10.1007/978-3-662-54458-7_8.
- 39 Fu Song and Zhilin Wu. On temporal logics with data variable quantifications: Decidability and complexity. *Inf. Comput.*, 251:104–139, 2016. doi:10.1016/j.ic.2016.08.002.
- 40 David Turner and Glynn Winskel. Nominal domain theory for concurrency. In *Computer Science Logic, CSL 2009*, pages 546–560, 2009. doi:10.1007/978-3-642-04027-6_39.

Test of Quantumness with Small-Depth Quantum Circuits

Shuichi Hirahara

National Institute of Informatics, Tokyo, Japan

François Le Gall

Graduate School of Mathematics, Nagoya University, Japan

Abstract

Recently Brakerski, Christiano, Mahadev, Vazirani and Vidick (FOCS 2018) have shown how to construct a test of quantumness based on the learning with errors (LWE) assumption: a test that can be solved efficiently by a quantum computer but cannot be solved by a classical polynomial-time computer under the LWE assumption. This test has led to several cryptographic applications. In particular, it has been applied to producing certifiable randomness from a single untrusted quantum device, self-testing a single quantum device and device-independent quantum key distribution.

In this paper, we show that this test of quantumness, and essentially all the above applications, can actually be implemented by a very weak class of quantum circuits: constant-depth quantum circuits combined with logarithmic-depth classical computation. This reveals novel complexity-theoretic properties of this fundamental test of quantumness and gives new concrete evidence of the superiority of small-depth quantum circuits over classical computation.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory

Keywords and phrases Quantum computing, small-depth circuits, quantum cryptography

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.59

Related Version *ArXiv Version*: <https://arxiv.org/abs/2105.05500>

Funding JSPS KAKENHI grants Nos. JP19H04066, JP20H05966, JP20H00579, JP20H04139, JP21H04879 and MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) grants No. JP-MXS0118067394 and JPMXS0120319794.

Acknowledgements The authors are grateful to Ryo Hiromasa, Tomoyuki Morimae, Yasuhiko Takahashi and Seiichiro Tani for helpful discussions.

1 Introduction

Background. A very active research area in quantum computing is proving the superiority of “weak” models of quantum computation, such as small-depth quantum circuits, over classical computation. The main motivation is that such models are expected to be much easier to implement than universal quantum computation (e.g., polynomial-size quantum circuits) – Indeed in the past years we have been witnessing the development of several small-scale quantum computers (see, e.g., [1] for information about current quantum computers).

Under assumptions such as the non-collapse of the polynomial hierarchy or the hardness of (appropriate versions of) the permanent, strong evidence of the superiority of weak classes of quantum circuits has been obtained from the 2000s [2, 3, 4, 6, 12, 13, 14, 18, 19, 20, 33, 39]. A recent breakthrough by Bravyi, Gosset and König [10], further strengthened by subsequent works [5, 11, 17, 21], showed an *unconditional* separation between the computational powers of quantum and classical small-depth circuits by exhibiting a computational task that can be solved by constant-depth quantum circuits but requires logarithmic depth for classical circuits. A major shortcoming, however, is that logarithmic-depth classical computation is a



© Shuichi Hirahara and François Le Gall;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 59; pp. 59:1–59:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

relatively weak complexity class. Due to the notorious difficulty of proving superlogarithmic lower bounds on the depth of classical circuits, showing significantly stronger unconditional separations seems completely out of reach of current techniques.

Progress has nevertheless been achieved recently by modifying the concept of computational problem, and considering *interactive problems* (problems consisting of several rounds of interaction between the computational device and a verifier). Grier and Schaeffer [23], in particular, showed that there exists an interactive problem that can be solved by constant-depth quantum circuits but such that any classical device solving it would solve $\oplus\text{L}$ -problems. This is a stronger evidence of the superiority of constant-depth quantum circuits since the complexity class $\oplus\text{L}$ is expected to be significantly larger than logarithmic-depth classical computation. On the other hand, problems in $\oplus\text{L}$ are still tractable classically since they can be solved in polynomial time.¹

Another significant development was achieved by Brakerski, Christiano, Mahadev, Vazirani and Vidick [7] who proposed, using some techniques from [29], a test of quantumness based on the Learning with Errors (LWE) assumption, which states that the learning with error problem (informally, inverting a “noisy” system of equations) cannot be solved in polynomial time. (See also [8, 27] for variants of this test.) They showed that this test can be passed with high probability using a polynomial-time quantum device but cannot be solved by any polynomial-time classical device under the LWE assumption, which is a compelling evidence of the superiority of quantum computing.² A crucial property of this test is that checking if the computational device passes the test (which thus means checking if the computational device is quantum) can be done efficiently – this property is not known to be true for many other tests from prior works in quantum supremacy (e.g., [2, 3, 4, 6, 12, 13, 14, 18, 19, 20, 33, 39].) Finally, the test of quantumness from [7] has another fundamental property: it can be shown that the only way for a computationally bounded quantum prover to pass the test is to prepare precisely the expected quantum state.³ This property makes it possible to control a computationally bounded quantum prover, and has already lead to many cryptographic applications: producing certifiable randomness from a single untrusted (computationally bounded) quantum device [7], self-testing of a single quantum device [31] and device-independent key distribution [30].

Our results. In this paper we investigate complexity-theoretic aspects of quantum protocols passing the above test of quantumness based on LWE. While the quantum protocol from [7] can clearly be implemented in polynomial time, and while prior works discussed its practical realization and gave some promising numerical estimates on the number of qubits needed for its implementation (for instance, Ref. [7] mentioned 2000 qubits for a protocol providing 50 bits of security), to our knowledge several theoretical aspects, and in particular depth complexity, have not been investigated so far.

We first isolate the main computational task solved by a quantum protocol passing the test. This computational problem, which we denote **StateGeneration**, is presented in Section 3. Informally, it asks to prepare a quantum superposition of an arbitrary vector x and its shift $x - s$, where s denotes the solution of the “noisy” system of linear equations used in the LWE assumption. Our main technical contribution (the formal statement is in Section 3) shows that this problem can be solved by a constant-depth quantum circuit combined with efficient (low-complexity) classical computation:

¹ More precisely, we have the inclusions $\text{NC}_1 \subseteq \text{L} \subseteq \oplus\text{L} \subseteq \text{NC}_2 \subseteq \text{P}$.

² We stress that the quantum protocol that passes the test does not solve the learning with error problem.

³ The proof of this statement relies on the (standard) assumption that the learning with error problem is hard for computationally bounded quantum computation as well.

► **Theorem 1** (Informal version). *The computational task `StateGeneration` can be solved by a constant-depth quantum circuit combined with logarithmic-depth classical computation.*

The model of quantum circuits we consider in Theorem 1 is described formally in Section 2.4 and is reminiscent of some models used in prior works on measurement-based quantum computing (in particular Refs. [15, 16]). The primary motivation for considering this model is as follows: compared with the practical cost of implementing quantum computation, classical computation (and especially low-complexity computation such as logarithmic-depth classical computation) can be considered as a free resource and thus may not be included in the depth complexity. One possible criticism of our model is that the quantum states created by our constant-depth quantum circuits need to be kept coherent while the logarithmic-depth classical computation is performed, which may be an issue since in terms of decoherence waiting is essentially as difficult as performing quantum computation. We can however argue that classical logarithmic-depth classical computation should be implementable significantly faster than logarithmic-depth quantum computation, thus limiting the impact of decoherence.

As mentioned above, `StateGeneration` is the main computational task used in the test of quantumness based on LWE and its applications given in [7, 8, 30, 31] (the other quantum steps indeed only consist in measuring the state generated in an appropriate basis). As a consequence of Theorem 1, the whole test of quantumness and its applications to producing certifiable randomness, self-testing and device-independent key distribution can thus immediately be implemented by constant-depth quantum circuits combined with logarithmic-depth classical computation. For completeness, we describe in detail how to apply our construction with the whole test of quantumness from [7], which was actually only sketched in prior works (since those works focused on applications of the test), in Section 4.

Overview of our techniques. Our main technical contribution is Theorem 1, which shows how to solve `StateGeneration` using constant-depth quantum circuits (in our model allowing some low-complexity classical pre/processing). This is done by modifying the construction of prior works in two major ways.

Our first contribution is to show how to construct in constant depth a quantum state robust against small “noise”. In [7] the construction was done by considering a state with amplitudes taken from a wide-enough Gaussian distribution, and creating this state using the approach from the seminal paper by Regev [36], which itself relied on a technique by Grover and Rudolph [24]. To our knowledge, the resulting construction, while definitely implementable with quantum circuits of polynomial size, does not seem to be implementable in constant depth. Instead, our main idea (see Theorem 9 in Section 3) is to use a quantum state with amplitudes taken from a much simpler distribution (a wide-enough truncated uniform distribution) that can be implemented in constant depth.

The second contribution (Theorem 6 in Section 3) is analyzing carefully how to implement in the quantum setting the map used in the learning with error problem (note that in the quantum setting the map needs to be applied in superposition, which requires a quantum circuit). We observe that when given as input a state robust against small noise, the remaining computational task involves only algebraic operations modulo q , for some large integer q . We then show that prior works by Høyer and Spalek [25] and Takahashi and Tani [38] imply that implementing arithmetic operations modulo q exactly and generating a good approximation of the uniform superposition of all elements of $\{0, 1, \dots, q - 1\}$ can be done using constant-depth quantum circuits if unbounded fanout gates are allowed. We finally show that unbounded fanout gates can be implemented in our model using a technique called gate teleportation [22, 28, 34].

2 Preliminaries

2.1 General notations

In this paper the notation \log represents the logarithm in basis 2. For any integer q , we write $\mathbb{Z}_q = \{0, 1, \dots, q-1\}$. As usual in lattice-based cryptography, we will often identify \mathbb{Z}_q with the set of integers $\{-\lfloor q/2 \rfloor + 1, \dots, \lfloor q/2 \rfloor\}$. For any $a \in \mathbb{Z}_q$, we write $J(a) \in \{0, 1\}^{\lceil \log q \rceil}$ its binary representation, as in [7]. Given a vector $x \in \mathbb{Z}_q^m$, we write $\|x\| = \sqrt{\sum_{i=1}^m |x_i|^2}$ and $\|x\|_\infty = \max_{i \in \{1, \dots, m\}} |x_i|$, and write $J(x) = (J(x_1), \dots, J(x_m)) \in \{0, 1\}^{m \lceil \log q \rceil}$ its binary representation. Given a matrix $A \in \mathbb{Z}_q^{m \times n}$, we define the distance of A as the minimum over all the non-zero vectors $x \in \mathbb{Z}_q^m$, of the quantity $\|Ax\|$.

2.2 Lattice-based cryptography

For a security parameter λ , let m, n, q be integer functions of λ . Let χ be a distribution over \mathbb{Z}_q . The $\text{LWE}_{m,n,q,\chi}$ problem is to distinguish between the distributions $(A, As + e)$ and (A, u) , where $A \in \mathbb{Z}_q^{m \times n}$, $s \in \mathbb{Z}_q^n$ and $u \in \mathbb{Z}_q^m$ are uniformly random and $e \leftarrow \chi^m$. The corresponding hardness assumption is that no polynomial-time algorithm can solve this problem with non-negligible advantage in λ . As in [7], we write $\text{LWE}_{n,q,\chi}$ the task of solving $\text{LWE}_{m,n,q,\chi}$ for any function m that is at most a polynomial in $n \log q$.

The most usual distribution χ used in lattice-based cryptography is the truncated discrete Gaussian distribution, which we now introduce. For any positive integer q and any positive real number B , the truncated discrete Gaussian distribution over \mathbb{Z}_q with parameter B , which we denote $D_{q,B}$, is defined as $D_{q,B}(x) = (e^{-\pi|x|^2/B^2})/\gamma$ if $|x| \leq B$ and $D_{q,B}(x) = 0$ otherwise, for any $x \in \mathbb{Z}_q$, where γ is the normalization factor defined as $\gamma = \sum_{z \in \mathbb{Z}_q, |z| \leq B} e^{-\pi|z|^2/B^2}$.

As in [7], we will use the following theorem to generate instances of the learning with error problem.

► **Theorem 2** (Theorem 2.6 in [7] and Theorem 5.1 in [32]). *Let $m, n \geq 1$ and $q \geq 2$ be such that $m = \Omega(n \log q)$. There is an efficient randomized algorithm $\text{GENTRAP}(1^n, 1^m, q)$ that returns a matrix $A \in \mathbb{Z}_q^{m \times n}$ and a trapdoor t_A such that the distribution of A is negligibly (in n) close to the uniform distribution. Moreover, there is an efficient algorithm INVERT that, on input A , t_A and $Ax + e$ where $x \in \mathbb{Z}_q^n$ is arbitrary, $\|e\| \leq q/(C\sqrt{n \log q})$ and C is a universal constant, returns x with overwhelming probability over $(A, t_A) \leftarrow \text{GENTRAP}(1^n, 1^m, q)$.*

The matrix A generated by $\text{GENTRAP}(1^n, 1^m, q)$ has distance at least $2q/(C\sqrt{n \log q})$ with overwhelming probability. Also note that if $\|e\|_\infty \leq q/(C\sqrt{mn \log q})$, then the inequality $\|e\| \leq q/(C\sqrt{n \log q})$ holds. These two observations motivate the following definition: we define \mathcal{K} as the set of 5-tuples (m, n, q, A, u) such that m, n and q are positive integers, $A \in \mathbb{Z}_q^{m \times n}$ is a matrix of distance at least $2q/(C\sqrt{n \log q})$, where C is the constant from Theorem 2, and $u \in \mathbb{Z}_q^m$ is a vector that can be written as $u = As + e$ for some $s \in \mathbb{Z}_q^n$ and some $e \in \mathbb{Z}_q^m$ with $\|e\|_\infty \leq q/(C\sqrt{mn \log q})$. Informally, the set \mathcal{K} represents the set of good parameters for the version of LWE we will consider. For technical reasons, we also define the following variant, which enables us to set a stronger upper bound on $\|e\|_\infty$. For any $B_V > 0$, we define $\mathcal{K}_{B_V} \subseteq \mathcal{K}$ as the set of 5-tuples $(m, n, q, A, u) \in \mathcal{K}$ such that the following two conditions hold:

- (i) $q \geq B_V C \sqrt{mn \log q}$,
- (ii) u can be written as $u = As + e$ for some $s \in \mathbb{Z}_q^n$ and some $e \in \mathbb{Z}_q^m$ with $\|e\|_\infty \leq B_V$.

2.3 Quantum states: bounded and robust states

We assume that the reader is familiar with the basics of quantum computing and refer to, e.g., [35] for a good reference.

For any positive integer q , we write \mathcal{H}_q the complex Hilbert space of dimension q with basis $\{|x\rangle\}_{x \in \mathbb{Z}_q}$. Quantum states in \mathcal{H}_q are (implicitly) implemented using $\lceil \log q \rceil$ qubits, via the binary encoding of these basis vectors. For any integer $m \geq 1$, we also consider the Hilbert space $\mathcal{H}_q^{\otimes m}$ and associate to it the basis $\{|x\rangle\}_{x \in \mathbb{Z}_q^m}$. A quantum state $|\varphi\rangle$ in $\mathcal{H}_q^{\otimes m}$ can thus be written as $|\varphi\rangle = \sum_{x \in \mathbb{Z}_q^m} \alpha_x |x\rangle$, for complex numbers α_x such that $\sum_{x \in \mathbb{Z}_q^m} |\alpha_x|^2 = 1$. We write its support $\text{supp}(|\varphi\rangle) = \{x \in \mathbb{Z}_q^m \mid \alpha_x \neq 0\}$. We say that $|\varphi\rangle$ has real amplitudes if $\alpha_x \in \mathbb{R}$ for each $x \in \mathbb{Z}_q^m$. For any vector $e \in \mathbb{Z}_q^m$, we write $|\varphi + e\rangle = \sum_{x \in \mathbb{Z}_q^m} \alpha_x |x + e\rangle$, where the addition is performed modulo q .

We now introduce two crucial definitions on which our approach will be based.⁴

► **Definition 3.** *Let B be a positive real number. A quantum state $|\varphi\rangle \in \mathcal{H}_q^{\otimes m}$ is B -bounded if $\|x\|_\infty < B$ for any element $x \in \text{supp}(|\varphi\rangle)$.*

► **Definition 4.** *Let B, ε be two positive real numbers. A quantum state $|\varphi\rangle \in \mathcal{H}_q^{\otimes m}$ is (ε, B) -robust if $|\varphi\rangle$ has real amplitudes and, for any vector $e \in \mathbb{Z}_q^m$ such that $\|e\|_\infty \leq B$, the inequality $\langle \varphi | \varphi + e \rangle \geq 1 - \varepsilon$ holds.*

Finally, given two states $|\varphi\rangle$ and $|\psi\rangle$ in $\mathcal{H}_q^{\otimes m}$, and any positive real number ε , we say that $|\varphi\rangle$ and $|\psi\rangle$ are ε -close if $\| |\varphi\rangle - |\psi\rangle \|^2 \leq \varepsilon$. We also define the notion of ε -closeness to a subspace as follows.

► **Definition 5.** *Let \mathcal{H}' be a subspace of $\mathcal{H}_q^{\otimes m}$ and ε be a positive real number. We say that a state $|\varphi\rangle \in \mathcal{H}_q^{\otimes m}$ is ε -close to \mathcal{H}' if there exists a quantum state $|\psi\rangle \in \mathcal{H}'$ such that $\| |\varphi\rangle - |\psi\rangle \|^2 \leq \varepsilon$.*

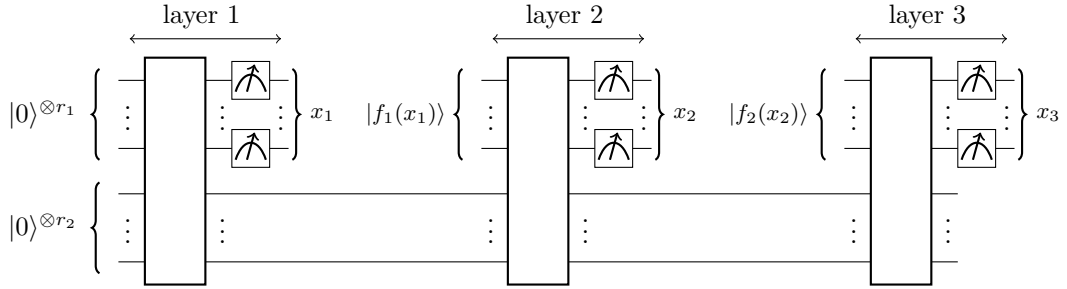
2.4 Quantum circuits

Universal sets of quantum gates. As in the standard model of quantum circuits (see, e.g., [35]), in this paper we work with qubits. We consider two sets of elementary gates. We first consider the set $\mathcal{B}_r = \{H, T, CNOT\}$ where $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is the Hadamard gate, $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$ is the $\pi/8$ -phase operation and $CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ is the controlled-not gate. This is an universal set consisting of a finite number of gates that can approximate any quantum gate with good precision (see Section 4.5.3 of [35] for details). The second set we consider, which we denote \mathcal{B} , contains all the gates acting on 1 qubit and the $CNOT$ operator. Note that this set contains an infinite number of gates.

Our model. We now introduce the class of quantum circuits considered in this paper. Let r_1 and r_2 be two positive integers, and \mathcal{S} be a set of elementary quantum gates (e.g., $\mathcal{S} = \mathcal{B}_r$ or $\mathcal{S} = \mathcal{B}$).

⁴ We stress that these two definitions (as well as several definitions of the previous paragraph) are basis-dependent – we always refer to the canonical basis $\{|x\rangle\}_{x \in \mathbb{Z}_q^m}$. Also note that while Definition 4 can easily be written without the requirement that the state has real amplitude (by replacing $\langle \varphi | \varphi + e \rangle$ by $|\langle \varphi | \varphi + e \rangle|$, for instance), requiring that the state has real amplitudes will be enough for our purpose and will simplify later calculations.

A circuit in the class $\mathcal{C}(\mathcal{S}, r_1, r_2)$ acts on $r_1 + r_2$ qubits. These qubits are initialized to the state $|0\rangle^{\otimes(r_1+r_2)}$. The circuit consists of successive layers. Each layer consists of a constant-depth quantum circuit over the basis \mathcal{S} acting on these $r_1 + r_2$ qubits, which does not contain any measurement, followed by measurements in the computational basis of all the first r_1 qubits. Consider the i -th layer. Let $x_i \in \{0, 1\}^{r_1}$ denote the outcome of measuring the first r_1 qubits at the end of this layer. Then some classical function $f_i: \{0, 1\}^{r_1} \rightarrow \{0, 1\}^{r_1}$ is applied to the x_i , and the value $f_i(x_i)$ is given as input to the first r_1 qubits of the next layer, i.e., the r_1 qubits are reinitialized to the state $|f_i(x_i)\rangle$. We refer to Figure 1 for an illustration.



■ **Figure 1** A quantum circuit of the class \mathcal{C} consisting of three layers. Each rectangular box represents a quantum circuit (without measurements) of constant depth with gates in the set \mathcal{S} .

The complexity of a circuit in the class defined above depends on the number of qubits $r_1 + r_2$, the number of layers and the classical complexity of computing function f_i 's. We are mainly interested in circuits that have a constant number of layers and such that all functions can be computed efficiently classically. We formally define this class below.

We define the class $\mathcal{C}_0(\mathcal{S})$ of families of circuits $\{C_n\}_{n \in \mathbb{N}}$ such that the following conditions hold:

- for each $n \in \mathbb{N}$, we have $C_n \in \mathcal{C}(\mathcal{S}, r_1, r_2)$ for some integers r_1, r_2 such that $r_1 + r_2 = n$;
- for each $n \in \mathbb{N}$, the number of layers in C_n is constant (i.e., independent of n);
- for each $n \in \mathbb{N}$, all the functions f_i 's of C_n can be computed by a $O(\log n)$ -depth classical circuit.

We require that the family is logarithmic-space uniform, i.e., there exists a classical Turing machine that on input 1^n outputs a classical description of C_n (as well as descriptions of the circuits computing the functions f_i 's) in $O(\log n)$ space.

2.5 Clifford circuits and quantum arithmetic

Clifford circuits. Let us consider the Pauli gates $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ and the phase gate $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$. A quantum circuit consisting only of gates from the set $\{X, Z, S, H, CNOT\}$ is called a Clifford circuit.⁵ Such a circuit can be implemented by a quantum circuit of class $\mathcal{C}_0(\mathcal{B}_r)$ acting on $\text{poly}(s)$ qubits, where s is the number of gates in the original circuit, via a technique called gate teleportation first introduced by Gottesman and Chuang [22] and then developed into a computational model by Leung [28] and Nielsen [34] (see also, e.g., [9, 26] for good presentations of this technique).

A concrete example, which we will actually heavily use, is the unbounded fanout gate over $\mathcal{H}_2^{\otimes m}$. This unitary gate maps the basis state $|x_1, x_2, \dots, x_{m-1}, x_m\rangle$ to $|x_1, x_1 \oplus x_2, \dots, x_1 \oplus x_{m-1}, x_1 \oplus x_m\rangle$, for any $x_1, \dots, x_m \in \{0, 1\}$. This gate can easily be written as a circuit

⁵ Since $X = S^2$ and $Z = HS^2H$, the two Pauli gates can actually be removed from this gate set.

consisting of $m - 1$ successive CNOT gates (the depth of such a circuit implementation is thus linear in m). Using the above approach, this gate can be implemented by a quantum circuit of class $\mathcal{C}_0(\mathcal{B}_r)$ acting on $\text{poly}(m)$ qubits. A concrete decomposition, which uses only two layers, is presented in Section 6 of [15].

Modular arithmetics. Let us consider the following unitary operations (where the arithmetic operations are performed modulo q and ω is a q -th root of unity):

- the quantum Fourier transform F_q over \mathcal{H}_q , such that $F_q|i\rangle = \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} \omega^{ij}|j\rangle$ for any $i \in \mathbb{Z}_q$;
- the unitary operation ADD_q over $\mathcal{H}_q^{\otimes 2}$ that maps $|i\rangle|j\rangle$ to $|i\rangle|i+j\rangle$ for any $i, j \in \mathbb{Z}_q$;
- the unitary operation MULT_q over $\mathcal{H}_q^{\otimes 3}$ that maps $|i\rangle|j\rangle|k\rangle$ to $|i\rangle|j\rangle|k+ij\rangle$ for any $i, j, k \in \mathbb{Z}_q$.

We now discuss how to obtain exact implementations for ADD_q and MULT_q , and also for arbitrary linear maps over \mathbb{Z}_q (exact implementation of these gates will be crucial for implementing our test of quantumness in constant depth). Takahashi and Tani [38] showed how to implement exactly ADD_q and MULT_q in constant depth by circuits that use gates in \mathcal{B} and unbounded fanout gates acting on $\text{poly}(\log q)$ qubits, by showing that quantum threshold gates, which are enough to implement all these operations (as first pointed out by Høyer and Spalek [25], based on prior works on classical threshold gates [37]), can be implemented in constant depth by such circuits. Since each unbounded fanout gate can be implemented by a quantum circuit of class $\mathcal{C}_0(\mathcal{B}_r)$ acting on $\text{poly}(\log q)$ qubits, as discussed above, these arithmetic operations can be exactly implemented by quantum circuits of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(\log q)$ qubits. As discussed in [25, 38], the same approach can be applied to implement iterated addition, and more generally any linear map $f: \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$, since such maps can be computed in constant depth using classical threshold gates as well. This can easily be further generalized to give implementation of any linear map $f: \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q^m$ by a quantum circuit of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(m, n, \log q)$ qubits.

Unfortunately, it is still unknown if the operator F_q can be implemented exactly in constant depth with a circuit using only elementary gates in \mathcal{B} and unbounded fanout gates (see Section 6 of [38]). For the protocol constructed in this paper, however, we will only need to apply F_q to the state $|0\rangle \in \mathcal{H}_q$, i.e., we only need to prepare the state $F_q|0\rangle = \frac{1}{\sqrt{q}} \sum_{x \in \mathbb{Z}_q} |x\rangle$. Lemma 4.18 in [25] shows that this task can be implemented in constant depth with exponential precision (which will be enough for our purpose): there exists a constant-depth circuit of size $\text{poly}(\log q)$ using gates in \mathcal{B} and unbounded fanout gates that computes a state which is at distance at most $1/q^2$ of the state $F_q|0\rangle$. By converting each unbounded fanout gate, this circuit can immediately be converted into a circuit in the class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(\log q)$ qubits.

3 Quantum State Generation using Small-Depth Circuits

In this section we describe the main computational task solved by a quantum prover in the test of quantumness based on LWE we present in Section 4 (as well as in prior works [7, 8, 30, 31]), and show how to solve it using a quantum circuit of small depth.

3.1 Statement of the problem

For any $B_V > 0$ and any $k = (m, n, q, A, u) \in \mathcal{K}_{B_V}$, where \mathcal{K}_{B_V} is the set of parameters defined in Section 2.2, let $\Lambda_k \subseteq \mathbb{Z}_q^m$ denote the set of vectors $y \in \mathbb{Z}_q^m$ such that there exists a vector $x \in \mathbb{Z}_q^n$ for which $\|Ax - y\| \leq q/(C\sqrt{n \log q})$. Note that such x is necessarily unique,

since A has distance at least $2q/(C\sqrt{n\log q})$. Let us write this vector x_y . Note that $x_u = s$ using the notations of Section 2.2, i.e., defining s as the (unique) vector in \mathbb{Z}_q^n such that u can be written as $u = As + e$ for $e \in \mathbb{Z}_q^m$ with $\|e\|_\infty \leq B_V$. For any $y \in \Lambda_k$, define the quantum state

$$|\Psi_y\rangle = \frac{1}{\sqrt{2}} (|0\rangle|x_y\rangle + |1\rangle|x_y - x_u\rangle).$$

Let \mathcal{H}_k be the subspace of $\mathcal{H}_2 \otimes \mathcal{H}_q^n \otimes \mathcal{H}_q^m$ generated by the states $\{|\Psi_y\rangle|y\rangle\}_{y \in \Lambda_k}$.

The computational problem we consider in this section, which we denote **StateGeneration**, has two parameters $\varepsilon, B_V > 0$, and is defined as follows. This is the main task solved by the quantum protocols passing our test of quantumness, as well as in the tests used in prior works [7, 8, 30, 31].

StateGeneration(ε, B_V).

Given $k \in \mathcal{K}_{B_V}$, create a quantum state ε -close to \mathcal{H}_k .

Here is our main theorem, which shows that the problem can be solved by a small-depth quantum circuit when q is large enough.

► **Theorem 1** (Formal version). *For any $\varepsilon, B_V > 0$, the problem **StateGeneration**(ε, B_V) can be solved, for all inputs $k \in \mathcal{K}_{B_V}$ such that $q \geq (8mB_VC\sqrt{mn\log q})/\varepsilon$, by a quantum circuit of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(m, n, \log q)$ qubits.*

Theorem 1 follows from Theorems 6 and 9 proved in Subsections 3.2 and 3.3.

3.2 Preparation procedure

In this subsection we present and analyze a quantum procedure that outputs a state close to \mathcal{H}_k when given as additional input an appropriate quantum state $|\varphi\rangle \in \mathcal{H}_q^{\otimes m}$. This procedure can be implemented by a small-depth quantum circuit. In subsection 3.3 we will show how to create efficiently such an appropriate state $|\varphi\rangle$.

The following theorem is the main contribution of this subsection.

► **Theorem 6.** *Let ε and B_V be any positive parameters. For any $k \in \mathcal{K}_{B_V}$ with $q \geq \sqrt{2n/\varepsilon}$, there exists a quantum circuit of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(m, n, \log q)$ qubits that receives a quantum state $|\varphi\rangle \in \mathcal{H}_q^{\otimes m}$, outputs a quantum state $|\Phi\rangle \in \mathcal{H}_2 \otimes \mathcal{H}_q^n \otimes \mathcal{H}_q^m$, and satisfies the following condition: if $|\varphi\rangle$ is $\frac{q}{C\sqrt{mn\log q}}$ -bounded and $(\varepsilon/2, B_V)$ -robust, then $|\Phi\rangle$ is ε -close to \mathcal{H}_k .*

Proof. We first describe the procedure. Let us write $|\varphi\rangle = \sum_{z \in \mathbb{Z}_q^m} \alpha_z |z\rangle$ the input state, where $\alpha_z \in \mathbb{R}$ for all $z \in \mathbb{Z}_q^m$ (remember that the definition of a robust state implies that the amplitudes are real). The procedure first prepares the state $|0\rangle|0\rangle|\varphi\rangle \in \mathcal{H}_2 \otimes \mathcal{H}_q^{\otimes n} \otimes \mathcal{H}_q^{\otimes m}$ and applies the unitary operator $H \otimes F_q^{\otimes n} \otimes I$ to this state to obtain

$$\frac{1}{\sqrt{2q^n}} \sum_{b \in \{0,1\}} \sum_{x \in \mathbb{Z}_q^n} |b\rangle|x\rangle|\varphi\rangle = \sum_{b \in \{0,1\}} \sum_{x \in \mathbb{Z}_q^n} \sum_{z \in \mathbb{Z}_q^m} \frac{\alpha_z}{\sqrt{2q^n}} |b\rangle|x\rangle|z\rangle.$$

Using the approach discussed in Section 2.5, this can be done by a quantum circuit of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(m, n, \log q)$ qubits with approximation error $\frac{n}{q^2} \leq \varepsilon/2$. Below we assume that this state has been done exactly – we will add the approximation error at the very end of the calculation. The procedure then converts this state to the state

$$|\Phi\rangle = \sum_{b \in \{0,1\}} \sum_{x \in \mathbb{Z}_q^n} \sum_{z \in \mathbb{Z}_q^m} \frac{\alpha_z}{\sqrt{2q^n}} |b\rangle|x\rangle|z + f_k(b, x)\rangle,$$

where $f_k: \{0, 1\} \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q^m$ is the function defined as $f_k(b, x) = Ax + bu$ for any $(b, x) \in \{0, 1\} \times \mathbb{Z}_q^n$ (all the operations are performed modulo q). This operation can be implemented by a quantum circuit of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(m, n, \log q)$ qubits using the approach of Section 2.5 since f_k can be written as a linear map over $\mathbb{Z}_q \times \mathbb{Z}_q^n$ as follows: define the matrix $A' \in \mathbb{Z}_q^{m \times (n+1)}$ obtained by appending the vector u to the left of the matrix A and write $f_k(b, x) = A' \begin{pmatrix} b \\ x \end{pmatrix}$.

We now analyze this procedure. Let us write the output state in the following form:

$$|\Phi\rangle = \frac{1}{\sqrt{2q^n}} \sum_{x \in \mathbb{Z}_q^n} (|0\rangle|x\rangle|\Phi_{0,x}\rangle + |1\rangle|x\rangle|\Phi_{1,x}\rangle),$$

where

$$|\Phi_{0,x}\rangle = \sum_{z \in \mathbb{Z}_q^m} \alpha_z |Ax + z\rangle \quad \text{and} \quad |\Phi_{1,x}\rangle = \sum_{z \in \mathbb{Z}_q^m} \alpha_z |Ax + u + z\rangle = \sum_{z \in \mathbb{Z}_q^m} \alpha_z |A(x + s) + e + z\rangle.$$

Define the quantum state

$$|\Phi'\rangle = \frac{1}{\sqrt{2q^n}} \sum_{x \in \mathbb{Z}_q^n} (|0\rangle|x\rangle|\Phi'_{0,x}\rangle + |1\rangle|x\rangle|\Phi'_{1,x}\rangle),$$

where $|\Phi'_{0,x}\rangle = |\Phi_{0,x}\rangle$ and $|\Phi'_{1,x}\rangle = \sum_{z \in \mathbb{Z}_q^m} \alpha_z |A(x + s) + z\rangle$. We first show that the states $|\Phi\rangle$ and $|\Phi'\rangle$ are close.

▷ **Claim 7.** $\langle \Phi | \Phi' \rangle \geq 1 - \varepsilon/4$.

Proof. We have $u = As + e$ for some $s \in \mathbb{Z}_q^n$ and some vector $e \in \mathbb{Z}_q^m$ such that $\|e\|_\infty \leq B_V$. Since the state $|\varphi\rangle$ is $(\varepsilon/2, B_V)$ -robust, we thus have $\langle \Phi_{1,x} | \Phi'_{1,x} \rangle = \langle \varphi | \varphi + e \rangle \geq 1 - \varepsilon/2$ for any $x \in \mathbb{Z}_q^n$. We thus obtain $\langle \Phi | \Phi' \rangle = \frac{1}{2} + \frac{1}{2q^n} \sum_{x \in \mathbb{Z}_q^n} \langle \Phi_{1,x} | \Phi'_{1,x} \rangle \geq 1 - \varepsilon/4$, as claimed. ◀

We now show that the state $|\Phi'\rangle$ is in \mathcal{H}_k . The crucial property we will use is that the equality $|\Phi'_{0,x}\rangle = |\Phi'_{1,x-s}\rangle$ holds for any $x \in \mathbb{Z}_q^n$.

Let us decompose $|\Phi'\rangle$ as follows:

$$|\Phi'\rangle = \sum_{y \in \mathbb{Z}_q^m} \gamma_y |\Phi'_y\rangle |y\rangle,$$

for quantum states $|\Phi'_y\rangle$ and amplitudes γ_y such that $\sum_{y \in \mathbb{Z}_q^m} |\gamma_y|^2 = 1$. We now show the following claim.

▷ **Claim 8.** For any $y \in \mathbb{Z}_q^m$ such that $|\gamma_y| > 0$, we have $y \in \Lambda_k$ and $|\Phi'_y\rangle = |\Psi_y\rangle$.

Proof. Assume that $|\gamma_y| > 0$. Observe that in this case $y \in \text{supp}(|\Phi'_{0,x_0}\rangle)$ for some $x_0 \in \mathbb{Z}_q^n$. Since the state $|\varphi\rangle$ is $q/(C\sqrt{mn \log q})$ -bounded, we have $\|y - Ax_0\| \leq \sqrt{m} \cdot \|y - Ax_0\|_\infty \leq q/(C\sqrt{n \log q})$, and thus $y \in \Lambda_k$.

We show below that for any distinct $x, x' \in \mathbb{Z}_q^n$ we have $\text{supp}(|\Phi'_{0,x}\rangle) \cap \text{supp}(|\Phi'_{0,x'}\rangle) = \emptyset$, which implies that $|\Phi'_y\rangle = |\Psi_y\rangle$.

Indeed, assume that $\text{supp}(|\Phi'_{0,x}\rangle) \cap \text{supp}(|\Phi'_{0,x'}\rangle) \neq \emptyset$ and take an element r in the intersection. Since the state $|\varphi\rangle$ is B_P -bounded, we have $\|r - Ax\| \leq \sqrt{m} \cdot \|r - Ax\|_\infty \leq q/(C\sqrt{n \log q})$ and $\|r - Ax'\| \leq \sqrt{m} \cdot \|r - Ax'\|_\infty \leq q/(C\sqrt{n \log q})$, and thus $\|A(x - x')\| \leq 2q/(C\sqrt{n \log q})$. This is impossible, since by construction the matrix A has distance at least $2q/(C\sqrt{n \log q})$. ◀

Claim 8 implies that the state $|\Phi'\rangle$ is in \mathcal{H}_k . Since we have $\| |\Phi\rangle - |\Phi'\rangle \|^2 = 2 - 2\langle \Phi | \Phi' \rangle \leq \varepsilon/2$ from Claim 7, this concludes the proof of the theorem (the additional $\varepsilon/2$ term comes from the approximation error in the application of $F_q^{\otimes n}$). ◀

3.3 Creating the initial state

Brakerski et al. [7] have shown how to construct a quantum state that is B_P -bounded and (ε, B_V) -robust, for appropriate parameters $B_V \ll B_P$, using Gaussian distributions. In this subsection we present another quantum state that has similar properties, but can be created by a small-depth quantum circuit.

► **Theorem 9.** *For any $\varepsilon, B_V > 0$, any integer $m \geq 1$ and any $q \geq (8mB_VC\sqrt{mn\log q})/\varepsilon$, there exists a quantum circuit of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(m, \log q)$ qubits that generates a quantum state $|\varphi\rangle \in \mathcal{H}_q^{\otimes m}$ that is $\frac{q}{C\sqrt{mn\log q}}$ -bounded and $(\varepsilon/2, B_V)$ -robust.*

Proof. Let us write $r = \left\lceil \log_2 \left(\frac{q}{C\sqrt{mn\log q}} \right) \right\rceil$ and $I = \{-2^{r-1}, \dots, 0, \dots, 2^{r-1} - 1\}$.

We describe the construction. Starting with the quantum state $|0\rangle^{\otimes m} \in \mathcal{H}_q^{\otimes m}$, apply (in parallel) a Hadamard gate on the first r qubits of each copy of $|0\rangle$, in order to get the state

$$\left(\frac{1}{\sqrt{2^r}} \sum_{x \in \{0, \dots, 2^r - 1\}} |x\rangle \right)^{\otimes m}.$$

Then apply on each of the m copies the unitary operator over \mathcal{H}_q that maps $|i\rangle$ to $|i - 2^{r-1}\rangle$ for any $i \in \mathbb{Z}_q$ (the subtraction is done modulo q). As described in Section 2.5, these arithmetic operations can be implemented by a quantum circuit of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(m, \log q)$ qubits. This gives the state

$$\left(\frac{1}{\sqrt{2^r}} \sum_{x \in I} |x\rangle \right)^{\otimes m} = \frac{1}{\sqrt{2^{mr}}} \sum_{(x_1, \dots, x_m) \in I^m} |x_1, \dots, x_m\rangle.$$

For any vector $e = (e_1, \dots, e_m) \in \mathbb{Z}_q^m$, consider the state

$$|\varphi + e\rangle = \frac{1}{\sqrt{2^{mr}}} \sum_{(x_1, \dots, x_m) \in I^m} |x_1 + e_1, \dots, x_m + e_m\rangle.$$

The inner product of $|\varphi\rangle$ and $|\varphi + e\rangle$ is $\langle \varphi | \varphi + e \rangle = \frac{|S_e|}{2^{mr}}$, where S_e is the set of vectors $(x_1, \dots, x_m) \in I^m$ such that $(x_1 + e_1, \dots, x_m + e_m) \in I^m$. If $\|e\|_\infty \leq B_V$, then $\{-2^{r-1} + B_V, \dots, 2^{r-1} - 1 - B_V\}^m \subset S_e$ and thus

$$\langle \varphi | \varphi + e \rangle \geq \left(\frac{2^r - 2B_V}{2^r} \right)^m = \left(1 - \frac{B_V}{2^{r-1}} \right)^m \geq 1 - \frac{mB_V}{2^{r-1}} \geq 1 - \frac{4mB_VC\sqrt{mn\log q}}{q} \geq 1 - \varepsilon/2,$$

as claimed. ◀

4 Application: Test of Quantumness

In this section we describe and analyze the test of quantumness based on the LWE assumption that has been implicitly presented in [7], and show how to use the results from Section 3 to pass this test with small-depth quantum circuits.

We first define some sets $G_{s,b,x} \subseteq \{0, 1\}^{n \lceil \log q \rceil}$ exactly as in [7]. The definition is fairly technical and can actually be skipped on a first reading, since we will later only use the property that these sets are dense enough. For any $b \in \{0, 1\}$ and any $x \in \mathbb{Z}_q^n$, let $I_{b,x}: \{0, 1\}^{n \lceil \log q \rceil} \rightarrow \{0, 1\}^n$ be the map such that for any $d \in \{0, 1\}^{n \lceil \log q \rceil}$, each coordinate

of $I_{b,x}(d)$ is obtained by taking the inner product modulo 2 of the corresponding block of $\lceil \log q \rceil$ coordinates of d and of $J(x) \oplus J(x - (-1)^b \mathbf{1})$, where $\mathbf{1}$ denotes the vector in \mathbb{Z}_q^n where each coordinate is $1 \in \mathbb{Z}_q$. We define the set

$$G_{b,x} = \left\{ d \in \{0, 1\}^{n \lceil \log q \rceil} \mid \exists i \in \left\{ b \frac{n}{2}, \dots, b \frac{n}{2} + \frac{n}{2} \right\} : (I_{b,x}(d))_i \neq 0 \right\}.$$

For any $s \in \mathbb{Z}_q^m$, we then define $G_{s,0,x} = G_{0,x} \cap G_{1,x-s}$ and $G_{s,1,x} = G_{0,x+s} \cap G_{1,x}$. Note that these sets are dense: for any $s, x \in \mathbb{Z}_q^n$ and any $b \in \{0, 1\}$, we have $|G_{s,b,x}| \geq (1 - 2 \cdot 2^{-n \lceil \log q \rceil / 4}) 2^{n \lceil \log q \rceil}$.

Our test of quantumness is described in Figure 2. In Subsection 4.1 we explain how to pass the test when q is large enough using a quantum prover that can be implemented in constant depth. In Subsection 4.2 we then show that no classical computationally-bounded prover can pass this test with high probability under the LWE assumption, for a large range of parameters. A concrete test of quantumness can be obtained, for instance, by fixing $\varepsilon = 1/n$, setting $B_L = \Theta(n)$, $m = \Theta(n^2)$, choosing B_V superpolynomial in n and taking $q = \Theta(B_V n^{9/2})$. Theorem 10 shows that a small-depth quantum prover can pass the corresponding test of quantumness with probability close to $1 - 1/n$, while Theorem 11 shows that no polynomial-time classical prover can pass the test with probability significantly larger than $3/4$, under the LWE assumption (the gap between the success probabilities of classical and quantum provers can easily be further amplified using parallel repetitions).

Input: three positive integers m, n, q such that $q \geq B_V C \sqrt{mn \log q}$ holds.

1. The verifier applies the procedure $\text{GENTRAP}(1^n, 1^m, q)$ and gets a pair (A, t_A) . The verifier then takes a vector $s \in \mathbb{Z}_q^n$ uniformly at random, and a vector $e \in \mathbb{Z}_q^m$ by sampling each coordinate independently according to the distribution D_{q, B_V} . The verifier sends the pair $(A, As + e)$ to the prover.
2. The prover sends a vector $y \in \mathbb{Z}_q^m$ to the verifier.
3. The verifier chooses a random bit r uniformly at random and sends it to the prover.
4. If $r = 0$ then the prover sends a pair $(b, x) \in \{0, 1\} \times \mathbb{Z}_q^n$ to the verifier. If $r = 1$ then the prover sends a pair $(c, d) \in \{0, 1\} \times \{0, 1\}^{n \lceil \log q \rceil}$ to the verifier.
5. If $r = 0$ then the verifier accepts if and only if $\|Ax + bu - y\| \leq 2q / (C \sqrt{n \log q})$. If $r = 1$, then the verifier applies the procedure $\text{INVERT}(A, t_A, y)$ and get an output that we denote $x_0 \in \mathbb{Z}_q^n$. The verifier accepts if and only if the three conditions $\|Ax_0 - y\| \leq 2q / (C \sqrt{n \log q})$, $c = d \cdot (J(x_0) \oplus J(x_0 - s))$ and $d \in G_{s,0,x_0}$ all hold.

■ **Figure 2** Test of quantumness. Here $B_V > 0$ is a parameter.

4.1 Quantum protocol

Here is the main result of this subsection.

► **Theorem 10.** *Let ε and B_V be any positive parameters. There exists a quantum prover, which can be implemented by a circuit of class $\mathcal{C}_0(\mathcal{B})$ acting on $\text{poly}(m, n, \log q)$ qubits, that passes the test of Figure 2 with probability at least $1 - 3\sqrt{\varepsilon} - \delta$ for all values (m, n, q) such that $q \geq (8m B_V C \sqrt{mn \log q}) / \varepsilon$, where δ is some negligible function of the parameters.*

Proof. The 5-tuple $(m, n, q, A, As + e)$ is in \mathcal{K}_{B_V} with overwhelming probability (see the discussion after Theorem 2 in Section 2.2). We describe the quantum protocol under this assumption. After receiving the key at Step 1, the prover creates a state $|\varphi\rangle$ that is

59:12 Test of Quantumness with Small-Depth Quantum Circuits

$q/(C\sqrt{mn\log q})$ -bounded and $(\varepsilon/2, B_V)$ -robust using Theorem 9. Then the prover applies Theorem 6 using the state $|\varphi\rangle$ as input, which gives a state $|\Phi\rangle$ that is ε -close to some state in \mathcal{H}_k .

Let us first describe and analyze the remaining of the protocol under the assumption that $|\Phi\rangle$ is in \mathcal{H}_k (instead of being only close to \mathcal{H}_k). The prover measures the rightmost register of $|\Phi\rangle$. Let $y \in \mathbb{Z}_q^m$ denote the measurement outcome. The state after the measurement is

$$|\Psi_y\rangle = \frac{1}{\sqrt{2}} (|0\rangle|x_0\rangle + |1\rangle|x_0 - s\rangle) |y\rangle,$$

where $x_0 \in \mathbb{Z}_q^n$ is such that $\|Ax_0 - y\| \leq q/(C\sqrt{n\log q})$. At Step 2, the prover sends this value y . At Step 4, if the prover received $r = 0$, it measures the first two registers of the above state in the computational basis and simply sends to the verifier the measurement outcome (b, x) . This passes the verifier's check at Step 5 with certainty, since $\|Ax_0 - y\| \leq q/(C\sqrt{n\log q})$ and $A(x_0 - s) + u = Ax_0 + e$, with

$$\|Ax_0 + e - y\| \leq q/(C\sqrt{n\log q}) + \|e\| \leq q/(C\sqrt{n\log q}) + B_V\sqrt{m} \leq 2q/(C\sqrt{n\log q}).$$

If the prover received $r = 1$, it first applies an Hamadard gate on each qubit of the first two registers, which gives the state

$$\left(\frac{1}{2\sqrt{2^n}} \sum_{c \in \{0,1\}} \sum_{d \in \{0,1\}^n} \left((-1)^{J(x_0) \cdot d} + (-1)^{J(x_0 - s) \cdot d + c} \right) |c\rangle|d\rangle \right) |y\rangle.$$

The prover then measures the first two registers, and sends to the verifier the outcome (c, d) . Since (c, d) necessary satisfies the equality $J(x_0) \cdot d \equiv J(x_0 - s) \cdot d + c \pmod{2}$, and $d \in G_{s,0,x_0}$ with overwhelming probability due to the density of $G_{s,0,x_0}$, the verifier's check succeeds at Step 5 with overwhelming probability, i.e., probability at least $1 - \delta$ for some negligible function δ .

Since the actual state $|\Phi\rangle$ is only ε -close to \mathcal{H}_k (instead of being in \mathcal{H}_k as we assumed so far), using the triangular inequality we can conclude that the success probability on the actual state is at least $1 - \delta - \varepsilon - 2\sqrt{\varepsilon} \geq 1 - \delta - 3\sqrt{\varepsilon}$. ◀

4.2 Classical hardness

In this subsection we will use exactly the same parameters and hardness assumption as in [7].

Let λ be a security parameter. All the other parameters are functions of λ . Let q be a prime. Let $\ell, n, m \geq 1$ be polynomially bounded functions of λ , and B_L, B_V be positive integers such that the following conditions hold:

- $n = \Omega(\ell \log q)$ and $m = \Omega(n \log q)$,
- $2\sqrt{n} \leq B_L < B_V \leq q$,
- B_V/B_L is superpolynomial in λ .

Here is the main result of this subsection.

► **Theorem 11.** *Assume a choice of parameters as above. Assume the hardness assumption $\text{LWE}_{\ell,q,D_q,B_L}$ holds. No polynomial-time classical prover can pass the test of Figure 2 with probability greater than $3/4 + \mu$, for some negligible function μ of the security parameter λ .*

Proof. Consider a classical prover that passes the test with probability at least $3/4 + \mu$ for some function μ .

Let us write w the contents of the prover's memory and computation history at the end of Step 2 (note that y can be recovered from w). Let $\mathcal{A}_0(w)$ be the algorithm the prover applies when it receives 0 at Step 3, and $\mathcal{A}_1(w)$ be the algorithm the prover applies when it receives 1. Let consider the following strategy: Apply $\mathcal{A}_0(w)$ to get (b, x) , then rewind the computation and apply $\mathcal{A}_1(w)$ to get (c, d) , and finally output the 4-tuple (b, x, d, c) .

Let $p_0(w)$ denote the probability that the output of $\mathcal{A}_0(w)$ satisfies $\|Ax + bu - y\| \leq 2q/(C\sqrt{n \log q})$, and $p_1(w)$ denote the probability that the output of $\mathcal{A}_1(w)$ satisfies $c = d \cdot (J(x_0) \oplus J(x_0 - s))$ and $d \in G_{s,0,x_0}$. Our assumption implies that $\mathbb{E}_w[p_0(w)/2 + p_1(w)/2] \geq 3/4 + \mu$. Thus the overall probability that $\|Ax + bu - y\| \leq 2q/(C\sqrt{n \log q})$, $c = d \cdot (J(x_0) \oplus J(x_0 - s))$ and $d \in G_{s,0,x_0}$ all hold is at least

$$\mathbb{E}_w[1 - (1 - p_0(w)) - (1 - p_1(w))] = \mathbb{E}_w[(p_0(w) + p_1(w)) - 1] \geq 1/2 + 2\mu.$$

In this case we have $x_0 = x$ if $b = 0$ and $x_0 = x + s$ if $b = 1$, and thus $c = d \cdot (J(x) \oplus J(x - (-1)^b s))$ holds in both cases. Lemma 4.7 in [7], which we state for completeness in Appendix A, guarantees that μ must be negligible. ◀

References

- 1 List of quantum processors. URL: https://en.wikipedia.org/wiki/List_of_quantum_processors.
- 2 Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 333–342, 2011. doi:10.1145/1993636.1993682.
- 3 Scott Aaronson and Alex Arkhipov. BosonSampling is far from uniform. *Quantum Information & Computation*, 14(15-16):1383–1423, 2014. doi:10.26421/QIC14.15-16-7.
- 4 Scott Aaronson and Lijie Chen. Complexity-theoretic foundations of quantum supremacy experiments. In *Proceedings of the 32nd Computational Complexity Conference*, pages 22:1–22:67, 2017. doi:10.4230/LIPIcs.CCC.2017.22.
- 5 Adam Bene Watts, Robin Kothari, Luke Schaeffer, and Avishay Tal. Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 515–526, 2019.
- 6 Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. “Quantum supremacy” and the complexity of random circuit sampling. In *Proceedings of the 10th Innovations in Theoretical Computer Science conference*, pages 15:1–15:2, 2019. arXiv:1803.04402. doi:10.4230/LIPIcs.ITCS.2019.15.
- 7 Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science*, pages 320–331, 2018. Full version available as arXiv:1804.00640. doi:10.1109/FOCS.2018.00038.
- 8 Zvika Brakerski, Venkata Koppula, Umesh V. Vazirani, and Thomas Vidick. Simpler proofs of quantumness. In *Proceedings of the 15th Conference on the Theory of Quantum Computation, Communication and Cryptography*, volume 158 of *LIPIcs*, pages 8:1–8:14, 2020. doi:10.4230/LIPIcs.TQC.2020.8.
- 9 Zvika Brakerski and Henry Yuen. Quantum garbled circuits, 2020. arXiv:2006.01085.
- 10 Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. *Science*, 362(6412):308–311, 2018. doi:10.1126/science.aar3106.
- 11 Sergey Bravyi, David Gosset, Robert König, and Marco Tomamichel. Quantum advantage with noisy shallow circuits in 3d. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science*, pages 995–999, 2019. doi:10.1109/FOCS.2019.00064.
- 12 Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 467(2126):459–472, 2010. doi:10.1098/rspa.2010.0301.

- 13 Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. *Physical Review Letters*, 117:080501, 2016. doi:10.1103/PhysRevLett.117.080501.
- 14 Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum circuits. *Quantum*, 1:8, 2017. doi:10.22331/q-2017-04-25-8.
- 15 Anne Broadbent and Elham Kashefi. Parallelizing quantum circuits. *Theoretical Computer Science*, 410(26):2489–2510, 2009. doi:10.1016/j.tcs.2008.12.046.
- 16 Dan E. Browne, Elham Kashefi, and Simon Perdrix. Computational depth complexity of measurement-based quantum computation. In *Proceedings of the 5th Conference on Theory of Quantum Computation, Communication, and Cryptography*, volume 6519 of *Lecture Notes in Computer Science*, pages 35–46, 2010. doi:10.1007/978-3-642-18073-6_4.
- 17 Matthew Coudron, Jalex Stark, and Thomas Vidick. Trading locality for time: Certifiable randomness from low-depth circuits. *Communications of Mathematical Physics*, 2021. doi:10.1007/s00220-021-03963-w.
- 18 Edward Farhi and Aram W. Harrow. Quantum supremacy through the quantum approximate optimization algorithm, 2016. arXiv:1602.07674.
- 19 Keisuke Fujii, Hirotada Kobayashi, Tomoyuki Morimae, Harumichi Nishimura, Shuhei Tamate, and Seiichiro Tani. Impossibility of classically simulating one-clean-qubit model with multiplicative error. *Physical Review Letters*, 120:200502, 2018. doi:10.1103/PhysRevLett.120.200502.
- 20 Keisuke Fujii and Shuhei Tamate. Computational quantum-classical boundary of noisy commuting quantum circuits. *Scientific Reports*, 6(25598), 2016. doi:10.1038/srep25598.
- 21 François Le Gall. Average-case quantum advantage with shallow circuits. In *Proceedings of the 34th Computational Complexity Conference*, volume 137 of *LIPICs*, pages 21:1–21:20, 2019. doi:10.4230/LIPICs.CCC.2019.21.
- 22 Daniel Gottesman and Isaac L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402:390–393, 1999. doi:10.1038/46503.
- 23 Daniel Grier and Luke Schaeffer. Interactive shallow Clifford circuits: quantum advantage against NC^1 and beyond. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 875–888, 2020. doi:10.1145/3357713.3384332.
- 24 Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002. arXiv:quant-ph/0208112.
- 25 Peter Høyer and Robert Spalek. Quantum fan-out is powerful. *Theory of Computing*, 1(1):81–103, 2005. doi:10.4086/toc.2005.v001a005.
- 26 Richard Jozsa. An introduction to measurement based quantum computation, 2005. arXiv:quant-ph/0508124.
- 27 Gregory D. Kahanamoku-Meyer, Soonwon Choi, Umesh V. Vazirani, and Norman Y. Yao. Classically-verifiable quantum advantage from a computational Bell test, 2021. arXiv:2104.00687.
- 28 Debbie W. Leung. Quantum computation by measurements. *International Journal of Quantum Information*, 2(1):33–43, 2004. doi:10.1142/S0219749904000055.
- 29 Urmila Mahadev. Classical verification of quantum computations. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science*, pages 259–267, 2018. doi:10.1109/FOCS.2018.00033.
- 30 Tony Metger, Yfke Dulek, Andrea Coladangelo, and Rotem Arnon-Friedman. Device-independent quantum key distribution from computational assumptions. ArXiv:2010.04175 (Presented as a contributed talk at QIP’21), 2020. arXiv:2010.04175.
- 31 Tony Metger and Thomas Vidick. Self-testing of a single quantum device under computational assumptions. In *Proceedings of the 12th Innovations in Theoretical Computer Science Conference*, volume 185 of *LIPICs*, pages 19:1–19:12, 2021. doi:10.4230/LIPICs.ITCS.2021.19.

- 32 Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012. doi:10.1007/978-3-642-29011-4_41.
- 33 Tomoyuki Morimae, Keisuke Fujii, and Joseph F. Fitzsimons. Hardness of classically simulating the one-clean-qubit model. *Physical Review Letters*, 112:130502, 2014. doi:10.1103/PhysRevLett.112.130502.
- 34 Michael A. Nielsen. Quantum computation by measurement and quantum memory. *Physical Letters A*, 308:96–100, 2003. doi:10.1016/S0375-9601(02)01803-0.
- 35 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- 36 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. doi:10.1145/1568318.1568324.
- 37 Kai-Yeung Siu, Jehoshua Bruck, Thomas Kailath, and Thomas Hofmeister. Depth efficient neural networks for division and related problems. *IEEE Transactions on Information Theory*, 39(3):946–956, 1993. doi:10.1109/18.256501.
- 38 Yasuhiro Takahashi and Seiichiro Tani. Collapse of the hierarchy of constant-depth exact quantum circuits. *Computational Complexity*, 25(4):849–881, 2016. doi:10.1007/s00037-016-0140-0.
- 39 Barbara M. Terhal and David P. DiVincenzo. Adaptive quantum computation, constant depth quantum circuits and Arthur-Merlin games. *Quantum Information & Computation*, 4(2):134–145, 2004. doi:10.26421/QIC4.2-5.

A The Adaptive Hardcore Bit Lemma

For completeness, we reproduce below the statement of the adaptive hardcore bit lemma from [7] on which the proof of Theorem 11 is based.

► **Lemma 12** (Lemma 4.7 in [7]). *Assume a choice of parameters as in Section 4.2. Assume the hardness assumption $\text{LWE}_{\ell,q,D_q,B_L}$ holds. Let $s \in \{0,1\}^n$. Write*

$$H_s = \{(b, x, d, d \cdot (J(x) \oplus J(x - (-1)^b s)) \mid b \in \{0,1\}, x \in \mathbb{Z}_q^n, d \in G_{s,b,x}\}$$

$$\overline{H}_s = \{(b, x, d, c) \mid (b, x, d, c \oplus 1) \in H_s\}.$$

Consider a pair $(A, As + e)$ generated as follows: generate A using $\text{GENTRAP}(1^n, 1^m, q)$, then take $s \in \{0,1\}^n$ uniformly at random and e by sampling each coordinate independently according to the distribution $D_{\mathbb{Z}_q, B_V}$. Then for any polynomial-time algorithm \mathcal{A} that receives as input the pair $(A, As + e)$ there exists a negligible function $\mu(\lambda)$ such that

$$\left| \Pr[\mathcal{A}(A, As + e) \in H_s] - \Pr[\mathcal{A}(A, As + e) \in \overline{H}_s] \right| \leq \mu(\lambda).$$

On Search Complexity of Discrete Logarithm

Pavel Hubáček  

Charles University, Prague, Czech Republic

Jan Václavek 

Charles University, Prague, Czech Republic

Abstract

In this work, we study the discrete logarithm problem in the context of TFNP – the complexity class of search problems with a syntactically guaranteed existence of solutions for all instances. Our main results establish that suitable variants of the discrete logarithm problem are complete for the complexity class PPP, respectively PWPP, i.e., the subclasses of TFNP capturing total search problems with a solution guaranteed by the pigeonhole principle, respectively the weak pigeonhole principle. Besides answering an open problem from the recent work of Sotiraki, Zampetakis, and Zirdelis (FOCS’18), our completeness results for PPP and PWPP have implications for the recent line of work proving conditional lower bounds for problems in TFNP under cryptographic assumptions. In particular, they highlight that any attempt at basing average-case hardness in subclasses of TFNP (other than PWPP and PPP) on the average-case hardness of the discrete logarithm problem must exploit its structural properties beyond what is necessary for constructions of collision-resistant hash functions.

Additionally, our reductions provide new structural insights into the class PWPP by establishing two new PWPP-complete problems. First, the problem DOVE, a relaxation of the PPP-complete problem PIGEON. DOVE is the first PWPP-complete problem not defined in terms of an explicitly shrinking function. Second, the problem CLAW, a total search problem capturing the computational complexity of breaking claw-free permutations. In the context of TFNP, the PWPP-completeness of CLAW matches the known intrinsic relationship between collision-resistant hash functions and claw-free permutations established in the cryptographic literature.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases discrete logarithm, total search problems, completeness, TFNP, PPP, PWPP

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.60

Related Version *Full Version:* <https://arxiv.org/pdf/2107.02617>

Funding *Pavel Hubáček:* Supported by the Grant Agency of the Czech Republic under the grant agreement no. 19-27871X and by the Charles University projects PRIMUS/17/SCI/9 and UNCE/SCI/004.

Acknowledgements We wish to thank the anonymous reviewers for their helpful suggestions on the presentation of our results. The first author is grateful to Chethan Kamath for multiple enlightening discussions about TFNP and for suggesting DOVE as a name for a total search problem contained in the complexity class PPP.

1 Introduction

The discrete logarithm problem (DLP) and, in particular, its conjectured average-case hardness lies at the foundation of many practical schemes in modern cryptography. To day, no significant progress towards a generic efficient algorithm solving DLP has been made (see, e.g., the survey by Joux, Odlyzko, and Pierrot [17] and the references therein).

One of the distinctive properties of DLP is its *totality*, i.e., given a generator g of a cyclic group (G, \star) , we know that a solution x for DLP exists for any target element $t = g^x$ in the group. Thus, the perceived hardness of DLP does not stem from the uncertainty whether a



© Pavel Hubáček and Jan Václavek;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 60; pp. 60:1–60:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

solution exists but pertains to the search problem itself. In this respect, DLP is not unique – there are various total search problems with unresolved computational complexity in many domains such as algorithmic game theory, computation number theory, and combinatorial optimization, to name but a few. More generally, the complexity of all total search problems is captured by the complexity class TFNP.

In order to improve our understanding of the seemingly disparate problems in TFNP, Papadimitriou [20] suggested to classify total search problems based on syntactic arguments ensuring the existence of a solution. His approach proved to be extremely fruitful and it gave rise to various subclasses of TFNP that cluster many important total search problems. For example,

PPAD: formalizes parity arguments on directed graphs and captures, e.g., the complexity of computing Nash equilibria in bimatrix games [7, 3].

PPA: formalizes parity arguments on *undirected* graphs and captures, e.g., the complexity of Necklace splitting [9].

PPP: formalizes the pigeonhole principle and captures, e.g., the complexity of solving problems related to integer lattices [22].

PWPP: formalizes the *weak* pigeonhole principle and captures, e.g., the complexity of breaking collision-resistant hash functions and solving problems related to integer lattices [22].

DLP and TFNP

DLP seems to naturally fit the TFNP landscape. Though, a closer look reveals a subtle issue regarding its totality stemming from the need to certify that the given element g is indeed a generator of the considered group (\mathbb{G}, \star) or, alternatively, that the target element t lies in the subgroup of (\mathbb{G}, \star) generated by g . If the order $s = |\mathbb{G}|$ of the group (\mathbb{G}, \star) is known then there are two natural approaches. The straightforward approach would be to simply allow additional solutions in the form of distinct $x, y \in [s] = \{0, \dots, s-1\}$ such that $g^x = g^y$. By the pigeonhole principle, either $t = g^x$ for some $x \in [s]$ or there exists such a non-trivial collision $x, y \in [s]$. The other approach would be to leverage the Lagrange theorem that guarantees that the order of any subgroup must divide the order of the group itself. If we make the factorization of the order s of the group a part of the instance then it can be efficiently tested whether g is indeed a generator.

Despite being a prominent total search problem, DLP was not extensively studied in the context of TFNP so far. Only recently, Sotiraki, Zampetakis, and Zirdelis [22] presented a total search problem motivated by DLP. They showed that it lies in the complexity class PPP and asked whether it is complete for the complexity class PPP.

1.1 Our Results

In this work, we study formalizations of DLP as a total search problem and prove new completeness results for the classes PPP and PWPP.

Our starting point is the discrete logarithm problem in “general groups” suggested by Sotiraki et al. [22]. Given the order $s \in \mathbb{Z}$, $s > 1$, we denote by $\mathbb{G} = [s] = \{0, \dots, s-1\}$ the canonical representation of a set with s elements. Any efficiently computable binary operation on \mathbb{G} can be represented by a Boolean circuit $f: \{0, 1\}^l \times \{0, 1\}^l \rightarrow \{0, 1\}^l$ that evaluates the operation on binary strings of length $l = \lceil \log(s) \rceil$ representing the elements of \mathbb{G} . Specifically, the corresponding binary operation \star on \mathbb{G} can be computed by first taking the binary representation of the elements $x, y \in \mathbb{G}$, evaluating f on the resulting strings, and mapping the value back to \mathbb{G} . Note that the binary operation \star induced on \mathbb{G} by f in this way might not satisfy the group axioms and, thus, we refer to (\mathbb{G}, \star) as the induced *groupoid* adopting the terminology for a set with a binary operation common in universal algebra.

Assuming that (\mathbb{G}, \star) is a cyclic group, we might be provided with the representations of the identity element $id \in \mathbb{G}$ and a generator $g \in \mathbb{G}$, which, in particular, enable us to efficiently access the group elements via an indexing function $\mathcal{I}_{\mathbb{G}}: [s] \rightarrow \mathbb{G}$ computed as the corresponding powers of g (e.g. via repeated squaring). An instance of a general DLP is then given by a representation (s, f) inducing a groupoid (\mathbb{G}, \star) together with the identity element $id \in \mathbb{G}$, a generator $g \in \mathbb{G}$, and a target $t \in \mathbb{G}$; a solution for the instance (s, f, id, g, t) is either an index $x \in [s]$ such that $\mathcal{I}_{\mathbb{G}}(x) = t$ or a pair of distinct indices $x, y \in [s]$ such that $\mathcal{I}_{\mathbb{G}}(x) = \mathcal{I}_{\mathbb{G}}(y)$. Note that the solutions corresponding to non-trivial collisions in $\mathcal{I}_{\mathbb{G}}$ ensure totality of the instance irrespective of whether the induced groupoid (\mathbb{G}, \star) satisfies the group axioms – the indexing function $\mathcal{I}_{\mathbb{G}}$ either has a collision or it is a bijection and must have a preimage for any t .

The general DLP as defined above can clearly solve DLP in specific groups with efficient representation such as any multiplicative group \mathbb{Z}_p^* of integers modulo a prime p , which are common in cryptographic applications. On the other hand, it allows for remarkably unstructured instances and the connection to DLP is rather loose – as we noted above, the general groupoid (\mathbb{G}, \star) induced by the instance might not be a group, let alone cyclic. Therefore, we refer to this search problem as INDEX (see Definition 15 in Section 4 for the formal definition).

A priori, the exact computational complexity of INDEX is unclear. Sotiraki et al. [22] showed that it lies in the class PPP by giving a reduction to the PPP-complete problem PIGEON, where one is asked to find a preimage of the 0^n string or a non-trivial collision for a function from $\{0, 1\}^n$ to $\{0, 1\}^n$ computed by a Boolean circuit given as an input. No other upper or lower bound on INDEX was shown in [22]. Given that DLP can be used to construct collision-resistant hash functions [6], it seems natural to ask whether INDEX lies also in the class PWPP, a subclass of PPP defined by the canonical problem COLLISION, where one is asked to find a collision in a shrinking function computed by a Boolean circuit given as an input.

However, a closer look at the known constructions of collision-resistant hash functions from DLP reveals that they crucially rely on the homomorphic properties of the function $g^x = \mathcal{I}_{\mathbb{G}}(x)$. Given that (\mathbb{G}, \star) induced by an arbitrary instance of INDEX does not necessarily possess the structure of a cyclic group, the induced indexing function $\mathcal{I}_{\mathbb{G}}$ is not guaranteed to have any homomorphic properties and it seems unlikely that INDEX could be reduced to any PWPP-complete problem such as COLLISION. In Section 4, we establish that the above intuition is indeed correct since our Theorem 1 shows that INDEX is PPP-complete:

► **Theorem 1.** *INDEX is PPP-complete.*

On the other hand, we show that, by introducing additional types of solutions in the INDEX problem, we can enforce sufficient structure on the induced groupoid (\mathbb{G}, \star) that allows for a reduction to the PWPP-complete problem COLLISION. First, we add a solution type witnessing that the coset of t is not the whole \mathbb{G} , i.e., that $\{t \star a \mid a \in \mathbb{G}\} \neq \mathbb{G}$, which cannot be the case in a group. Specifically, a solution is also any pair of distinct $x, y \in [s]$ such that $t \star \mathcal{I}_{\mathbb{G}}(x) = t \star \mathcal{I}_{\mathbb{G}}(y)$. Second, we add a solution enforcing some form of homomorphism in $\mathcal{I}_{\mathbb{G}}$ with respect to t . Specifically, a solution is also any pair of $x, y \in [s]$ such that $\mathcal{I}_{\mathbb{G}}(x) = t \star \mathcal{I}_{\mathbb{G}}(y)$ and $\mathcal{I}_{\mathbb{G}}(x - y \bmod s) \neq t$. The second type of a solution is motivated by the classical construction of a collision-resistant hash function from DLP by Damgård [6]. Notice that if there are no solutions of the second type then any pair x, y such that $\mathcal{I}_{\mathbb{G}}(x) = t \star \mathcal{I}_{\mathbb{G}}(y)$ gives rise to the preimage of t under $\mathcal{I}_{\mathbb{G}}$ by simply computing $x - y \bmod s$. We refer to the version of INDEX with the additional two types of solutions as DLOG (see Definition 5 in Section 3 for the formal definition), as it is in our opinion close enough to the standard DLP in cyclic groups.

Since DLOG is a relaxation of INDEX obtained by allowing additional types of solutions, it could be the case that we managed to reduce DLOG to COLLISION simply because DLOG is trivial. Note that this is not the case since DLOG is at least as hard as DLP in any cyclic group with an efficient representation, where DLP would naturally give rise to an instance of DLOG with a unique solution corresponding to the solution for the DLP. In Section 3, we establish that DLOG is at least as hard as the problem of finding a non-trivial collision in a shrinking function by proving Theorem 2 that shows that DLOG is PWPP-complete:

► **Theorem 2.** *DLOG is PWPP-complete.*

Implications for cryptographic lower bounds for subclasses of TFNP

It was shown already by Papadimitriou [20] that cryptographic hardness might serve as basis for arguing the existence of average-case hardness in subclasses of TFNP. A recent line of work attempts to show such cryptographic lower bounds for subclasses of TFNP under increasingly more plausible cryptographic hardness assumptions [16, 2, 10, 13, 11, 18, 4, 5, 8, 1, 19, 15]. However, it remains an open problem whether DLP can give rise to average-case hardness in subclasses of TFNP other than PWPP and PPP. Our results highlight that any attempt at basing average-case hardness in subclasses of TFNP (other than PWPP and PPP) on the average-case hardness of the discrete logarithm problem must exploit its structural properties beyond what is necessary for constructions of collision-resistant hash functions.

Witnessing totality of number theoretic problems

In the full version [12], we discuss some of the issues that arise when defining total search problems corresponding to actual problems in computational number theory. First, we highlight some crucial distinctions between the general DLOG as defined in Definition 5 and the discrete logarithm problem in multiplicative groups \mathbb{Z}_p^* . In particular, we argue that the latter is unlikely to be PWPP-complete.

Second, we clarify the extent to which our reductions exploit the expressiveness allowed by the representations of instances of DLOG and INDEX. In particular, both the reduction from COLLISION to DLOG and from PIGEON to INDEX output instances that induce groupoids unlikely to satisfy group axioms and, therefore, do not really correspond to DLP. Additionally, we revisit the problem BLICHFELDT introduced in [22] and show that it also exhibits a similar phenomenon in the context of computational problems on integer lattices.

Alternative characterizations of PWPP

Our PWPP-completeness result for DLOG is established via a series of reductions between multiple intermediate problems, which are thus also PWPP-complete. We believe this characterization will prove useful in establishing further PWPP-completeness results. These new PWPP-complete problems are defined in Section 3 and an additional discussion is provided in Section 5.

2 Preliminaries

We denote by $[m]$ the set $\{0, 1, \dots, m-1\}$, by \mathbb{Z}^+ the set $\{1, 2, 3, \dots\}$ of positive integers, and by \mathbb{Z}_0^+ the set $\{0, 1, 2, \dots\}$ of non-negative integers. For two strings $u, v \in \{0, 1\}^*$, $u \parallel v$ stands for the concatenation of u and v . When it is clear from the context, we omit the operator \parallel , e.g., we write $0x$ instead of $0 \parallel x$. The standard XOR function on binary strings of equal lengths is denoted by \oplus .

Bit composition and decomposition

Throughout the paper, we often make use of the bit composition and bit decomposition functions between binary strings of length k and the set $[2^k]$ of non-negative integers less than 2^k . We denote these functions bc^k and bd^k . Concretely, $\text{bc}^k : \{0, 1\}^k \rightarrow [2^k]$ and $\text{bd}^k : [2^k] \rightarrow \{0, 1\}^k$. Formally, for $x = x_1x_2 \dots x_k \in \{0, 1\}^k$, we define $\text{bc}^k(x) = \sum_{i=0}^{k-1} x_{k-i}2^i$. The function bc^k is bijective and we define the function bd^k as its inverse, i.e., for $a \in [2^k]$, $\text{bd}^k(a)$ computes the unique binary representation of a with leading zeroes such that its length is k . When clear from the context, we omit k and write simply bc and bd to improve readability. At places, we work with the output of bd^k without the leading zeroes. We denote by $\text{bd}_0 : \mathbb{Z}_0^+ \rightarrow \{0, 1\}^*$ the standard function which computes the binary representation without the leading zeroes.

TFNP and some of its subclasses

A *total NP search problem* is a relation $S \subseteq \{0, 1\}^* \times \{0, 1\}^*$ such that: 1) the decision problem whether $(x, y) \in S$ is computable in polynomial-time in $|x| + |y|$, and 2) there exists a polynomial q such that for all $x \in \{0, 1\}^*$, there exists a $y \in \{0, 1\}^*$ such that $(x, y) \in S$ and $|y| \leq q(|x|)$. The class of all total NP search problems is denoted by TFNP.

Let $S, T \subseteq \{0, 1\}^* \times \{0, 1\}^*$ be total search problems. A *reduction from S to T* is a pair of polynomial-time computable functions $f, g : \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that, for all $x, y \in \{0, 1\}^*$ if $(f(x), y) \in T$ then $(x, g(y)) \in S$. In case there exists a reduction from S to T , we say that *S is reducible to T* . The above corresponds to so-called polynomial-time many-one (or Karp) reductions among decision problems in the context of search problems. In the rest of the paper, we consider only such reductions.

► **Definition 3** (PIGEON problem and PPP [20]).

INSTANCE: A Boolean circuit C with n inputs and n outputs.

SOLUTION: One of the following:

1. a string $u \in \{0, 1\}^n$ such that $C(u) = 0^n$,
2. distinct strings $u, v \in \{0, 1\}^n$ such that $C(u) = C(v)$.

The class of all total search problems reducible to PIGEON is called PPP.

► **Definition 4** (COLLISION problem and PWPP [16]).

INSTANCE: A Boolean circuit C with n inputs and m outputs with $m < n$.

SOLUTION: Distinct strings $u, v \in \{0, 1\}^n$ such that $C(u) = C(v)$.

The class of all total search problems reducible to COLLISION is called PWPP.

3 DLog is PWPP-complete

In this section, we define DLOG, a total search problem associated to DLP and show that it is PWPP-complete. Our reductions give rise to additional new PWPP-complete problems DOVE and CLAW, which we discuss further in Section 5.

Similarly to Sotiraki et al. [22], we represent a binary operation on $\mathbb{G} = [s] = \{0, \dots, s-1\}$ by a Boolean circuit $f : \{0, 1\}^l \times \{0, 1\}^l \rightarrow \{0, 1\}^l$, where $l = \lceil \log(s) \rceil$. Given such a representation (s, f) , we define a binary operator $f_{\mathbb{G}} : [s] \times [s] \rightarrow [2^l]$ for all $x, y \in [s]$ as $f_{\mathbb{G}}(x, y) = \text{bc}(f(\text{bd}(x), \text{bd}(y)))$. We denote by (\mathbb{G}, \star) the groupoid induced by f , where $\star : [s] \times [s] \rightarrow [s]$ is the binary operation closed on $[s]$ obtained by extending the operator $f_{\mathbb{G}}$ in some fixed way, e.g., by defining $x \star y = 1$ for all $x, y \in [s]$ such that $f_{\mathbb{G}}(x, y) \notin [s]$.

■ **Algorithm 1** Computation of the x -th power of the generator $g \in [s]$ of a groupoid (\mathbb{G}, \star) of size $s \in \mathbb{Z}_0^+$ induced by $f: \{0, 1\}^{2^{\lceil \log(s) \rceil}} \rightarrow \{0, 1\}^{\lceil \log(s) \rceil}$ with identity $id \in [s]$.

```

1: procedure  $\mathcal{I}_{\mathbb{G}}(x)$ 
2:    $(x_m, \dots, x_1) \leftarrow \text{bd}_0(x)$ 
3:    $r \leftarrow \text{bd}(id)$ 
4:    $g \leftarrow \text{bd}(g)$ 
5:   for  $i$  from  $m$  to 1 do
6:      $r \leftarrow f(r, r)$ 
7:     if  $x_i = 1$  then
8:        $r \leftarrow f(g, r)$ 
9:     end if
10:  end for
11:  return  $\text{bc}(r)$ 
12: end procedure

```

If the induced groupoid (\mathbb{G}, \star) was a cyclic group then we could find the indices of the identity element $id \in [s]$ and a generator $g \in [s]$. Moreover, we could use g to index the elements of the group (\mathbb{G}, \star) , e.g., in the order of increasing powers of g , and the corresponding *indexing function* $\mathcal{I}_{\mathbb{G}}: [s] \rightarrow [2^l]$ would on input x return simply the x -th power of the generator g . We fix a canonical way of computing the x -th power using the standard square-and-multiply method as defined in Algorithm 1. The algorithm first computes $(x_m, x_{m-1}, \dots, x_1) = \text{bd}_0(x)$, i.e., the binary representation of the exponent x without the leading zeroes for some $m \leq l$, and it then proceeds with the square-and-multiply method using the circuit f . As explained above, f implements the binary group operation. Hence, $f(r, r)$ corresponds to squaring the intermediate value r and $f(g, r)$ corresponds to multiplication of the intermediate value r by the generator g .

With the above notation in place, we can give the formal definition of DLOG.

► **Definition 5** (DLOG problem).

INSTANCE: A size parameter $s \in \mathbb{Z}^+$ such that $s \geq 2$ and a Boolean circuit $f: \{0, 1\}^{2^{\lceil \log(s) \rceil}} \rightarrow \{0, 1\}^{\lceil \log(s) \rceil}$ representing a groupoid (\mathbb{G}, \star) , and indices $id, g, t \in [s]$.

SOLUTION: One of the following:

1. $x \in [s]$ such that $\mathcal{I}_{\mathbb{G}}(x) = t$,
2. $x, y \in [s]$ such that $f_{\mathbb{G}}(x, y) \geq s$,
3. $x, y \in [s]$ such that $x \neq y$ and $\mathcal{I}_{\mathbb{G}}(x) = \mathcal{I}_{\mathbb{G}}(y)$,
4. $x, y \in [s]$ such that $x \neq y$ and $f_{\mathbb{G}}(t, \mathcal{I}_{\mathbb{G}}(x)) = f_{\mathbb{G}}(t, \mathcal{I}_{\mathbb{G}}(y))$,
5. $x, y \in [s]$ such that $\mathcal{I}_{\mathbb{G}}(x) = f_{\mathbb{G}}(t, \mathcal{I}_{\mathbb{G}}(y))$ and $\mathcal{I}_{\mathbb{G}}(x - y \bmod s) \neq t$.

The first type of a solution in DLOG corresponds to the discrete logarithm of t . Since we cannot efficiently verify that the input instance represents a group with the purported generator g , additional types of a solution had to be added in order to guarantee that DLOG is total. Note that any solution of these additional types witnesses that the instance does not induce a group, since for a valid group these types cannot happen. Nevertheless, the first three types of a solution are sufficient to guarantee the totality of DLOG. The last two types of a solution make DLOG to lie in the class PWPP and are crucial for correctness of our reduction from DLOG to COLLISION presented in Section 3.2. In Section 5, we provide further discussion of DLOG and some possible alternative definitions.

In Section 3.1, we show that DLOG is PWPP-hard. In Section 3.2, we show that DLOG lies in PWPP. Therefore, we prove Theorem 2.

► **Theorem 2.** *DLOG is PWPP-complete.*

3.1 DLog is PWPP-hard

To show that DLOG is PWPP-hard, we reduce to it from the PWPP-complete problem COLLISION (see Definition 4). Given an instance $C: \{0, 1\}^n \rightarrow \{0, 1\}^{n-1}$ of COLLISION, our reduction to DLOG defines a representation (s, f) of a groupoid (\mathbb{G}, \star) and the elements id, g , and t such that we are able to extract some useful information about C from any non-trivial collision $\mathcal{I}_{\mathbb{G}}(x) = \mathcal{I}_{\mathbb{G}}(y)$ in the indexing function $\mathcal{I}_{\mathbb{G}}$ computed by Algorithm 1. The main obstacle that we need to circumvent is that, even though the computation performed by $\mathcal{I}_{\mathbb{G}}$ employs the circuit f representing the binary operation in the groupoid, it has a very restricted form. In particular, we need to somehow define f using C so that there are no collisions in $\mathcal{I}_{\mathbb{G}}$ unrelated to solutions of the instance of COLLISION. To sidestep some of the potential issues when handling an arbitrary instance of COLLISION, we reduce to DLOG from an intermediate problem we call DOVE.

► **Definition 6** (DOVE problem).

INSTANCE: A Boolean circuit C with n inputs and n outputs.

SOLUTION: One of the following:

1. a string $u \in \{0, 1\}^n$ such that $C(u) = 0^n$,
2. a string $u \in \{0, 1\}^n$ such that $C(u) = 0^{n-1}1$,
3. distinct strings $u, v \in \{0, 1\}^n$ such that $C(u) = C(v)$,
4. distinct strings $u, v \in \{0, 1\}^n$ such that $C(u) = C(v) \oplus 0^{n-1}1$.

It is immediate that DOVE is a relaxation of PIGEON (cf. Definition 3) with two additional new types of a solution – the cases 2 and 4 in the above definition. Similarly to case 1, case 2 corresponds to a preimage of a fixed element in the range. Case 4 corresponds to a pair of strings such that their images under C differ only on the last bit. Permutations for which it is computationally infeasible to find inputs with evaluations differing only on a prescribed index appeared in the work of Zheng, Matsumoto, and Imai [23] under the term *distinction-intractable* permutations. Zheng et al. showed that distinction-intractability is sufficient for collision-resistant hashing. Note that we employ distinction-intractability in a different way than [23]. In particular, their construction of collision-resistant hash from distinction-intractable permutations could be leveraged towards a reduction from DOVE to COLLISION (proving DOVE is contained in PWPP) – we use DOVE as an intermediate problem when reducing from COLLISION to DLOG (proving PWPP-hardness of DLOG). In the overview of the reduction from DOVE to DLOG below, we explain why distinction-intractability seems as a natural choice for our definition of DOVE.

Reducing Dove to DLog

Let $C: \{0, 1\}^n \rightarrow \{0, 1\}^{n-1}$ be an arbitrary instance of DOVE. Our goal is to construct an instance $G = (s, f, id, g, t)$ of DLOG such that any solution to G provides a solution to the original instance C of DOVE. The key step in the construction of G is a suitable choice of the circuit f since it defines both $\mathcal{I}_{\mathbb{G}}$ and $f_{\mathbb{G}}$. Our initial observation is that, by the definition of $\mathcal{I}_{\mathbb{G}}$ (Algorithm 1), the circuit f is only applied on specific types of inputs during the computation of $\mathcal{I}_{\mathbb{G}}(x)$. Specifically:

- In each loop, $f(r, r)$ is computed for some $r \in \{0, 1\}^*$. We denote f restricted to this type of inputs by f_0 , i.e., $f_0(r) = f(r, r)$.
- If the corresponding bit of x is one then $f(g, r)$ is computed with fixed $g \in \{0, 1\}^*$ and some $r \in \{0, 1\}^*$. We denote f restricted to this type of inputs by f_1 , i.e., $f_1(r) = f(g, r)$.

Hence, using the above notation, the computation of $\mathcal{I}_G(x)$ simply corresponds to an iterated composition of the functions f_0 and f_1 depending on the binary representation of x evaluated on id (e.g., $\mathcal{I}_G(\text{bc}(101)) = f_1 \circ f_0 \circ f_0 \circ f_1 \circ f_0(\text{bd}(id))$). Exploiting the observed structure of the computation of \mathcal{I}_G , our approach is to define f_0 and f_1 (i.e., the corresponding part of f) using the circuit C so that we can extract some useful information about C from any non-trivial collision $\mathcal{I}_G(x) = \mathcal{I}_G(y)$ (i.e., from a solution to DLOG, case 3).

The straightforward option is to set $f_0(r) = f_1(r) = C(r)$ for all $r \in \{0, 1\}^n$. Unfortunately, such an approach fails since for all distinct $u, v \in \{0, 1\}^n$ with Hamming weight l , there would be an easy to find non-trivial collision $x = \text{bc}(u)$ and $y = \text{bc}(v)$ of the form $\mathcal{I}_G(x) = \text{bc}(C^{n+l}(id)) = \mathcal{I}_G(y)$, which might not provide any useful information about the circuit C . Hence, we define f_0 and f_1 such that $f_0 \neq f_1$.

On a high level, we set $f_0(r) = C(r)$ and $f_1(r) = C(h(r))$ for some function $h: \{0, 1\}^n \rightarrow \{0, 1\}^n$ that is not the identity as in the flawed attempt above. Then, except for some special case, a non-trivial collision $\mathcal{I}_G(x) = \mathcal{I}_G(y)$ corresponds to the identity $C(C(u)) = C(h(C(v)))$ for some $u, v \in \{0, 1\}^n$, which are not necessarily distinct. In particular, if $C(u) \neq h(C(v))$ then the pair of strings $C(u), h(C(v))$ forms a non-trivial collision for C . Otherwise, we found a pair u, v such that $C(u) = h(C(v))$ that, for the choice $h(y) = y \oplus 0^{n-1}1$, translates into $C(u) = C(v) \oplus 0^{n-1}1$, i.e., a pair of inputs breaking distinction-intractability of C , and corresponds to the fourth type of a solution in DOVE. Finally, the second type of a solution in DOVE captures the special case when there is no pair u, v such that $C(C(u)) = C(h(C(v)))$. The formal reduction from DOVE to DLOG establishing Lemma 7 below is given in the full version [12].

► **Lemma 7.** *DOVE is reducible to DLOG.*

PWPP-hardness of Dove. Next, we show that, by introducing additional types of solutions into the definition of PIGEON, we do not make the corresponding search problem too easy – DOVE is at least as hard as any problem in PWPP. Our reduction from COLLISION to DOVE is rather syntactic and natural. In particular, it results in instances of DOVE with only one type of solutions corresponding to collisions of the original instance of COLLISION. For the formal proof, see the full version [12].

► **Lemma 8.** *COLLISION is reducible to DOVE.*

Lemma 7 and Lemma 8 imply PWPP-hardness of DLOG as stated in the corollary below.

► **Corollary 9.** *DLOG is PWPP-hard.*

3.2 DLog Lies in PWPP

In order to establish that DLOG lies in PWPP, we build on the existing cryptographic literature on constructions of collision-resistant hash functions from the discrete logarithm problem. Specifically, we mimic the classical approach by Damgård [6] to first construct a family of *claw-free permutations* based on DLP and then define a collision-resistant hash using the family of claw-free permutations.¹ Recall that a family of claw-free permutations is an efficiently sampleable family of pairs of permutations such that given a “random” pair

¹ In principle, it might be possible to adapt any alternative known construction of collision-resistant hash from DLP such as the one of Ishai, Kushilevitz, and Ostrovsky [14], which goes through the intermediate object of *homomorphic one-way commitments*. However, this would necessitate not only the corresponding changes in the definition of DLOG but also an alternative proof of its PWPP-hardness.

h_0 and h_1 of permutations from the family, it is computationally infeasible to find a *claw* for the two permutations, i.e., inputs u and v such that $h_0(u) = h_1(v)$. We formalize the corresponding total search problem, which we call **CLAW**, below.

► **Definition 10** (**CLAW problem**).

INSTANCE: Two Boolean circuits h_0, h_1 with n inputs and n outputs.

SOLUTION: One of the following:

1. two strings $u, v \in \{0, 1\}^n$ such that $h_0(u) = h_1(v)$,
2. two distinct strings $u, v \in \{0, 1\}^n$ such that $h_0(u) = h_0(v)$,
3. two distinct strings $u, v \in \{0, 1\}^n$ such that $h_1(u) = h_1(v)$.

The first type of a solution in **CLAW** corresponds to finding a claw for the pair of functions h_0 and h_1 . As we cannot efficiently certify that both h_0 and h_1 are permutations, we introduce the second and third type of solutions which witness that one of the functions is not bijective. In other words, the second and third type of solution ensure the totality of **CLAW**.

Similarly to [6], our high-level approach when reducing from **DLOG** to **COLLISION** is to first reduce from **DLOG** to **CLAW** and then from **CLAW** to **COLLISION**. Although, we cannot simply employ his analysis since we have no guarantee that 1) the groupoid induced by an arbitrary **DLOG** instance is a cyclic group and 2) that an arbitrary instance of **CLAW** corresponds to a pair of permutations. It turns out that the second issue is not crucial. It was observed by Russell [21] that the notion of claw-free *pseudopermutations* is sufficient for collision-resistant hashing. Our definition of **CLAW** corresponds exactly to the worst-case version of breaking claw-free pseudopermutations as defined by [21]. As for the first issue, we manage to provide a formal reduction from **DLOG** to **GENERAL-CLAW**, a variant of **CLAW** defined below.

► **Definition 11** (**GENERAL-CLAW problem**).

INSTANCE: Two Boolean circuits h_0, h_1 with n inputs and n outputs and $s \in \mathbb{Z}^+$ such that $1 \leq s < 2^n$.

SOLUTION: One of the following:

1. two strings $u, v \in \{0, 1\}^n$ such that $\text{bc}(u) < s$, $\text{bc}(v) < s$ and $h_0(u) = h_1(v)$,
2. two distinct strings $u, v \in \{0, 1\}^n$ such that $h_0(u) = h_0(v)$,
3. two distinct strings $u, v \in \{0, 1\}^n$ such that $h_1(u) = h_1(v)$,
4. a string $u \in \{0, 1\}^n$ such that $\text{bc}(u) < s$ and $\text{bc}(h_0(u)) \geq s$,
5. a string $u \in \{0, 1\}^n$ such that $\text{bc}(u) < s$ and $\text{bc}(h_1(u)) \geq s$.

The main issue that necessitates the introduction of additional types of a solution in the definition of **GENERAL-CLAW** (compared to **CLAW**) is that the possible solutions to an instance of **DLOG** are not from the whole domain $[2^n]$ but they must lie in $[s]$.

The formal reductions proving Lemma 12 and Lemma 13 below are presented in the full version [12]. The two lemmata establish Corollary 14, which concludes the proof of Theorem 2.

► **Lemma 12.** *DLOG is reducible to GENERAL-CLAW.*

► **Lemma 13.** *GENERAL-CLAW is reducible to COLLISION.*

► **Corollary 14.** *DLOG is contained in PWPP.*

4 Index is PPP-complete

In this section, we study the complexity of a more restricted version of DLOG that we call INDEX. In the definition of INDEX, we use the notation from Section 3 introduced for the definition of DLOG. In particular, the function \mathcal{I}_G is the same as defined in Algorithm 1.

► **Definition 15** (INDEX problem).

INSTANCE: A size parameter $s \in \mathbb{Z}^+$ such that $s \geq 2$ and a Boolean circuit $f: \{0, 1\}^{2^{\lceil \log(s) \rceil}} \rightarrow \{0, 1\}^{\lceil \log(s) \rceil}$ representing a groupoid (G, \star) and indices $g, id, t \in [s]$.

SOLUTION: One of the following:

1. $x \in [s]$, such that $\mathcal{I}_G(x) = t$,
2. $x, y \in [s]$, such that $x \neq y$ and $f_G(x, y) \geq s$,
3. $x, y \in [s]$, such that $x \neq y$ and $\mathcal{I}_G(x) = \mathcal{I}_G(y)$.

It is immediate that DLOG is a relaxation of INDEX due to the additional types of solutions. In Section 4.1, we show that INDEX is PPP-hard. In Section 4.2, we show that INDEX lies in PPP. Therefore, we prove PPP-completeness of INDEX.

► **Theorem 1.** *INDEX is PPP-complete.*

4.1 Index is PPP-hard

The formal reduction from the PPP-complete problem PIGEON to INDEX is arguably the most technical. Given an instance $C: \{0, 1\}^n \rightarrow \{0, 1\}^n$ of PIGEON, the main idea is to define an instance $G = (s, f, id, g, t)$ of INDEX such that the induced indexing function \mathcal{I}_G carefully “emulates” the computation of the circuit C – so that any solution to G provides a solution to the original instance C of PIGEON. In order to achieve this, we exploit the structure of the computation induced by \mathcal{I}_G in terms of evaluations of the circuit f representing the binary operation in the groupoid (G, \star) . Specifically, the computation of \mathcal{I}_G gives rise to a tree labeled by the values output by \mathcal{I}_G and structured by the two special types of calls to f (i.e., squaring the intermediate value or multiplying it by the generator). Our reduction constructs f inducing \mathcal{I}_G with the computation corresponding to a sufficiently large such tree so that its leaves can represent all the possible inputs for the instance C of PIGEON and the induced indexing function \mathcal{I}_G outputs the corresponding evaluation of C at each leaf. Moreover, for the remaining nodes in the tree, \mathcal{I}_G results in a bijection to ensure there are no additional solutions of the constructed instance of INDEX that would be unrelated to the original instance of PIGEON. Below, we provide additional details of the ideas behind the formal reduction given in the full version [12].

Similarly to the reduction from DOVE to DLOG, the key step in our construction of G is a suitable choice of the circuit f since it determines the function \mathcal{I}_G . Recall the notation for f_0 and f_1 introduced in the reduction from DOVE to DLOG, i.e., $f_0(r) = f(r, r)$ and $f_1(r) = f(g, r)$. We start by describing a construction of an induced groupoid (G, \star) independent of the instance C of PIGEON but which serves as a natural step towards our reduction.

Constructing bijective \mathcal{I}_G

Our initial goal in the first construction is to define f_0 and f_1 and the elements $id, g \in [s]$ such that \mathcal{I}_G is the identity function, i.e., such that $\mathcal{I}_G(a) = a$ for all $a \in [s]$. To this end, our key observation is that, for many pairs of inputs $a, b \in [s]$, the computation of $\mathcal{I}_G(b)$ includes the whole computation of $\mathcal{I}_G(a)$ as a prefix (see Algorithm 1), e.g., for all $a, b \in [s]$ such that

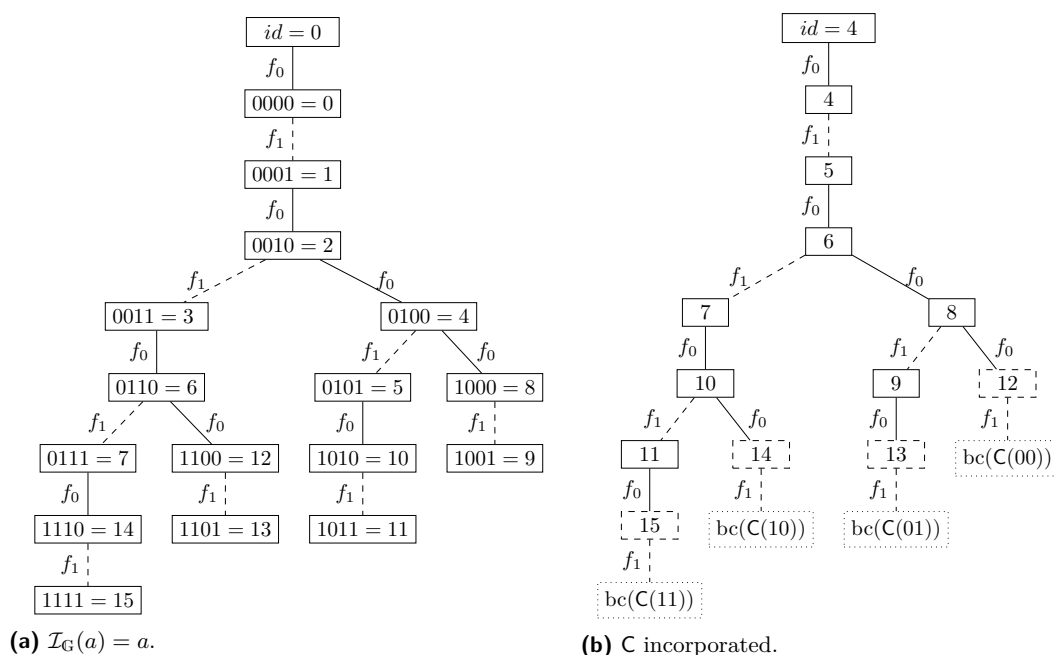


Figure 1 Trees induced by the computation of \mathcal{I}_G .

- either $bd_0(a)$ is a prefix of $bd_0(b)$
- or $bd_0(a) = y|0$ and $bd_0(b) = y|1$ for some $y \in \{0, 1\}^*$.

Specifically, if $bd_0(a) = y|0$ then $\mathcal{I}_G(a) = bc(f_0(bd(\mathcal{I}_G(bc(y)))))$, and if $bd_0(a) = y|1$ then $\mathcal{I}_G(a) = bc(f_1(bd(\mathcal{I}_G(bc(y|0)))))$.

Thus, we can capture the whole computation of \mathcal{I}_G on all the possible inputs from \mathbb{G} via a tree representing the successive calls to f_0 and f_1 based on the bit decomposition $bd_0(a)$ of the input a without the leading zeroes. In Figure 1a, we give a tree induced by the computation of \mathcal{I}_G in a groupoid of order $s = 16$ with $id = 0$. Solid lines correspond to the application of f_0 and dotted lines to application of f_1 . Except for the root labeled by the identity element id , each node of the tree corresponds to the point at which \mathcal{I}_G terminates on the corresponding input $a \in [s]$, where the second value in the label of the node is the input a and the first value is $bd(a)$, i.e., the binary representation of a with the leading zeroes.

Note that Figure 1a actually suggests which functions f_0 and f_1 induce \mathcal{I}_G such that $\mathcal{I}_G(a) = a$ for all $a \in [s]$. In particular, Algorithm 1 initializes the computation of \mathcal{I}_G with $r = bd(id) = bd(0) = 0^n$ and, thus, the desired traversal of the computation tree is achieved for all inputs $a \in [s]$ by 1) f_0 that performs a cyclic shift of the input r to the left and 2) f_1 that flips the last bit of the input r .

Similarly, the above observation allows to construct f'_0 and f'_1 such that for all $a \in [s]$ that $\mathcal{I}_G(a) = a + b \pmod s$ for some fixed $b \in [s]$, which can be performed simply by setting $id = b$ and consistently “shifting” the intermediate value r by the bit decomposition of the fixed value b before and after application of the above functions f_0 and f_1 .

Incorporating the Pigeon instance

The issue which makes it non-trivial to reduce from PIGEON to INDEX is that the functions f_0 and f_1 inducing the groupoid (\mathbb{G}, \star) are oblivious to the actual progress of the computation performed by \mathcal{I}_G . The above discussion shows that we have some level of control over the

60:12 On Search Complexity of Discrete Logarithm

computation of \mathcal{I}_G . However, it is a priori unclear how to meaningfully incorporate the PIGEON instance C into the above construction achieving that $\mathcal{I}_G(a) = a$ for all $a \in [s]$. For example, we cannot simply allow f_0 or f_1 to output $C(r)$ while at some internal node in the computation tree of \mathcal{I}_G as this would completely break the global structure of \mathcal{I}_G on the node and all its children and, in particular, could induce collisions in \mathcal{I}_G unrelated to the collisions in C . However, we can postpone the application of C to the leaves of the tree since, for all inputs a corresponding to a leaf in the tree, the computation of $\mathcal{I}_G(a)$ is not a part of the computation for $\mathcal{I}_G(b)$ for another input b .

Given that we are restricted to the leaves of the computation tree when embedding the computation of C into \mathcal{I}_G , we must work with a big enough tree in order to have as many leaves as the 2^n possible inputs of the circuit $C: \{0, 1\}^n \rightarrow \{0, 1\}^n$. In other words, the instance of INDEX must correspond to a groupoid of order s strictly larger than n . Note that for $s = 2^k$, the leaves of the tree correspond exactly to the inputs for \mathcal{I}_G from the set

$$A_o = \{a \in [2^k] \mid \exists y \in \{0, 1\}^{k-2}: \text{bd}(a) = 1\|y\|1\},$$

i.e., the set of *odd* integers between 2^{k-1} and 2^k , which has size 2^{k-2} . Thus, in our construction, we set $s = 2^{n+2}$ to ensure that there are 2^n leaves that can represent the domain of C .

Our goal is to define \mathcal{I}_G so that its restriction to the internal nodes of the tree (non-leaves) is a bijection between $[2^{n+2}] \setminus A_o$ and $[2^{n+2}] \setminus [2^n]$. In other words, when evaluated on any internal node of the tree, \mathcal{I}_G avoids the values in $[2^n]$ corresponding to bit composition of the elements in the range of C . If we manage to induce such \mathcal{I}_G then there are no non-trivial collisions in \mathcal{I}_G involving the internal nodes – the restrictions of \mathcal{I}_G to A_o and to its complement $[2^{n+2}] \setminus A_o$ would have disjoint images and, by the bijective property of the restriction to the internal nodes of the tree, any collision in \mathcal{I}_G would be induced by a collision in C . Our construction achieves this goal by starting with f_0 and f_1 inducing \mathcal{I}_G such that, for all $a \in [2^{n+2}]$, it holds that $\mathcal{I}_G(a) = a + 2^n \pmod{2^{n+2}}$, which we already explained above.

Note that the image of the restriction of \mathcal{I}_G to the set

$$A_e = \{a \in [2^{n+2}] \mid \exists y \in \{0, 1\}^n: \text{bd}(a) = 1\|y\|0\},$$

i.e., the set of *even* integers between 2^{n+1} and 2^{n+2} , has non-empty intersection with integers in $[2^n]$ corresponding to the range of C . Nevertheless, it is possible to locally alter the behaviour of f_0 and f_1 on A_e so that \mathcal{I}_G does not map to $[2^n]$ when evaluated on A_e . Then, we adjust the definition of f_0 and f_1 such that for all inputs $a \in A_o$ corresponding to a leaf of the tree, $\mathcal{I}_G(a) = \text{bc}(C(h(a)))$ for some bijection h between A_o and $\{0, 1\}^n$ (a natural choice is simply the function that drops the first and the last bit from the binary decomposition $\text{bd}(a)$ of a). Finally, we set the target in the resulting instance of INDEX to $t = 0$ to ensure that the preimage of t under \mathcal{I}_G corresponds exactly to a preimage of 0^n under C .

In Figure 1b, we illustrate the computation tree of \mathcal{I}_G corresponding to an instance of INDEX produced by our reduction on input $C: \{0, 1\}^n \rightarrow \{0, 1\}^n$ for $n = 2$. Accordingly, G is of size $s = 2^{n+2} = 16$ and its elements are represented by the nodes of the tree. When compared with the tree in Figure 1a, the label of each node in Figure 1b equals the value $\mathcal{I}_G(a)$, where a is the second value in the label of the node at the same position in the tree in Figure 1a. Nodes belonging to $[2^s] \setminus A_e \cup A_o$, A_e , and A_o are highlighted by differing styles of edges. Specifically, the labels of nodes with a solid edge correspond to evaluations of the inputs from $[2^{n+1}] = [8]$, the labels of nodes with a dashed edge correspond to evaluations of the inputs from A_e , and the labels of nodes with a dotted edge correspond to the evaluations

of the inputs from A_o . Since the image of $bc \circ C$ is $[2^n] = [4]$, it is straightforward to verify that any collision in \mathcal{I}_G depicted in Figure 1b must correspond to a collision in C and that any preimage of $t = 0$ under \mathcal{I}_G corresponds directly to a preimage of 0^n under C .

The formal reduction establishing Lemma 16 is given in the full version [12].

► **Lemma 16.** *PIGEON is reducible to INDEX.*

4.2 Index Lies in PPP

The main idea of our reduction from INDEX to PIGEON is analogous to the reduction in [22] from their discrete logarithm problem in “general groups” to PIGEON. Although, we need to handle the additional second type of a solution for INDEX, which corresponds to f_G outputting an element outside G . The formal reduction proving Lemma 17 is given in the full version [12]. Together, Lemma 16 and Lemma 17 establish Theorem 1.

► **Lemma 17.** *INDEX is reducible to PIGEON.*

5 New Characterizations of PWPP

Our results in Section 3.1 and Section 3.2 establish new PWPP-complete problems DLOG, DOVE, and CLAW. Below, we provide additional discussion of these new PWPP-complete problems.

DLog

Alternative types of violations. Since the last type of a solution in DLOG implies that the associative property does not hold for the elements $t, \mathcal{I}_G(x)$, and $\mathcal{I}_G(y)$, one could think about changing the last type of a solution to finding $x, y, z \in [s]$ such that $f_G(x, f_G(y, z)) \neq f_G(f_G(x, y), z)$ to capture violations of the associative property directly. However, our proof of PWPP-hardness would fail for such alternative version of DLOG and we do not see an alternative way of reducing to it from the PWPP-complete problem COLLISION. In more detail, any reduction from COLLISION to DLOG must somehow embed the instance C of COLLISION in the circuit f in the constructed instance of DLOG. However, a refutation of the associative property of the form $f(x, f(y, z)) \neq f(f(x, y), z)$ for some x, y , and z might simply correspond to a trivial statement $C(u) \neq C(v)$ for some $u \neq v$, which is unrelated to any non-trivial collision in C .

Explicit \mathcal{I}_G . A natural question about our definition of DLOG is whether its computational complexity changes if the instance additionally contains an explicit circuit computing the indexing function \mathcal{I}_G . First, the indexing function \mathcal{I}_G could then be independent of the group operation f and, thus, the reduction from COLLISION to such variant of DLOG would become trivial by defining the indexing function \mathcal{I}_G directly via the COLLISION instance C . On the other hand, the core ideas of the reduction from DLOG to COLLISION would remain mostly unchanged as it would have to capture also \mathcal{I}_G computed by Algorithm 1. Nevertheless, we believe that our version of DLOG with an implicit \mathcal{I}_G computed by the standard square-and-multiply algorithm strikes the right balance in terms of modeling an interesting problem. The fact that it is more structured than the alternative with an explicit \mathcal{I}_G makes it significantly less artificial and relevant to the discrete logarithm problem, which is manifested especially in the non-trivial reduction from DLOG to COLLISION in Section 3.2.

Dove

The chain of reductions in Section 3 shows, in particular, that DOVE (Definition 6) is PWPP-complete. The most significant property of DOVE compared to the known PWPP-complete problems (PIGEON or the weak constrained-SIS problem defined by Sotiraki et al. [22]) is that it is not defined in terms of an explicitly shrinking function. Nevertheless, it is equivalent to COLLISION and, thus, it inherently captures some notion of compression. Given its different structure than COLLISION, we were able to leverage it in our proof of PWPP-hardness of DLOG, and it might prove useful in other attempt at proving PWPP-hardness of other problems. We emphasize that all four types of a solution in DOVE are exploited towards our reduction from DOVE to DLOG and we are not aware of a more direct approach of reducing COLLISION to DLOG that avoids DOVE as an intermediate problem.

Claw

Russel [21] showed that a weakening of claw-free permutations is sufficient for collision-resistant hashing. Specifically, he leveraged claw-free *pseudopermutations*, i.e., functions for which it is also computationally infeasible to find a witness refuting their bijectivity. Our definition of CLAW ensures totality by an identical existential argument – a pair of functions with identical domain and range either has a claw or we can efficiently witness that one of the functions is not surjective.

CLAW trivially reduces to the PWPP-complete problem GENERAL-CLAW and, thus, it is contained in PWPP. Below, we provide also a reduction from COLLISION to CLAW establishing that it is PWPP-hard.

► **Lemma 18.** *COLLISION is reducible to CLAW.*

Proof of Lemma 18. We start with an arbitrary instance $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ of COLLISION with $m < n$. Without loss of generality, we can suppose that $m = n - 1$ since otherwise we can pad the output with zeroes, which preserves the collisions. We construct an instance of CLAW as follows:

$$h_0(x) = C(x)0$$

and

$$h_1(x) = C(x)1.$$

We show that any solution to this instance (h_0, h_1) of CLAW gives a solution to the original instance C of COLLISION. Three cases can occur:

1. $u, v \in \{0, 1\}^n$ such that $h_0(u) = h_1(v)$. Since the last bit of $h_0(u)$ is zero and the last bit of $h_1(v)$ is one, this case cannot happen.
2. $u, v \in \{0, 1\}^n$ such that $u \neq v$ and $h_0(u) = h_0(v)$. From the definition of h_0 , we get that $C(u)0 = h_0(u) = h_0(v) = C(v)0$, which implies that $C(u) = C(v)$. Hence, the pair u, v forms a solution to the original instance C of COLLISION.
3. $u, v \in \{0, 1\}^n$ such that $u \neq v$ and $h_1(u) = h_1(v)$. We can proceed analogously as in the previous case to show that the pair u, v forms a solution to the original instance C of COLLISION. ◀

References

- 1 Nir Bitansky and Idan Gerichter. On the cryptographic hardness of local search. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 6:1–6:29, 2020. doi:10.4230/LIPIcs.ITCS.2020.6.
- 2 Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1480–1498, 2015. doi:10.1109/FOCS.2015.94.
- 3 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J. ACM*, 56(3), 2009. doi:10.1145/1516512.1516516.
- 4 Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a Nash equilibrium is no easier than breaking Fiat-Shamir. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1103–1114. ACM, 2019. doi:10.1145/3313276.3316400.
- 5 Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. PPAD-hardness via iterated squaring modulo a composite. *IACR Cryptology ePrint Archive*, 2019:667, 2019. URL: <https://eprint.iacr.org/2019/667>.
- 6 Ivan Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology – EUROCRYPT ’87, Workshop on the Theory and Application of Cryptographic Techniques, Amsterdam, The Netherlands, April 13-15, 1987, Proceedings*, volume 304 of *Lecture Notes in Computer Science*, pages 203–216. Springer, 1987. doi:10.1007/3-540-39118-5_19.
- 7 Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009. doi:10.1137/070699652.
- 8 Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In *Advances in Cryptology – EUROCRYPT 2020 – 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 125–154, 2020. doi:10.1007/978-3-030-45727-3_5.
- 9 Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 638–649, 2019. doi:10.1145/3313276.3316334.
- 10 Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a Nash equilibrium. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 579–604, 2016. doi:10.1007/978-3-662-53008-5_20.
- 11 Pavel Hubáček, Moni Naor, and Eylon Yogev. The journey from NP to TFNP hardness. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 60:1–60:21, 2017. doi:10.4230/LIPIcs.ITCS.2017.60.
- 12 Pavel Hubáček and Jan Václavek. On search complexity of discrete logarithm. *CoRR*, abs/2107.02617, 2021. arXiv:2107.02617.
- 13 Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. *SIAM J. Comput.*, 49(6):1128–1172, 2020. doi:10.1137/17M1118014.
- 14 Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2005. doi:10.1007/978-3-540-30576-7_24.

- 15 Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. Snargs for bounded depth computations and PPAD hardness from sub-exponential LWE. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 708–721. ACM, 2021. doi:10.1145/3406325.3451055.
- 16 Emil Jeřábek. Integer factoring and modular square roots. *J. Comput. Syst. Sci.*, 82(2):380–394, 2016. doi:10.1016/j.jcss.2015.08.001.
- 17 Antoine Joux, Andrew M. Odlyzko, and Cécile Pierrot. The past, evolving present, and future of the discrete logarithm. In Çetin Kaya Koç, editor, *Open Problems in Mathematics and Computational Science*, pages 5–36. Springer, 2014. doi:10.1007/978-3-319-10683-0_2.
- 18 Ilan Komargodski and Gil Segev. From Minicrypt to Obfustopia via private-key functional encryption. *J. Cryptol.*, 33(2):406–458, 2020. doi:10.1007/s00145-019-09327-x.
- 19 Alex Lombardi and Vinod Vaikuntanathan. Fiat-Shamir for repeated squaring with applications to PPAD-hardness and VDFs. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 632–651. Springer, 2020. doi:10.1007/978-3-030-56877-1_22.
- 20 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.
- 21 Alexander Russell. Necessary and sufficient conditions for collision-free hashing. *J. Cryptol.*, 8(2):87–100, 1995. doi:10.1007/BF00190757.
- 22 Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. PPP-completeness with connections to cryptography. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 148–158. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00023.
- 23 Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. Duality between two cryptographic primitives. In Shojiro Sakata, editor, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 8th International Symposium, AAECC-8, Tokyo, Japan, August 20-24, 1990, Proceedings*, volume 508 of *Lecture Notes in Computer Science*, pages 379–390. Springer, 1990. doi:10.1007/3-540-54195-0_66.

A Homological Condition on Equational Unifiability

Mirai Ikebuchi ✉

Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

Equational unification is the problem of solving an equation modulo equational axioms. In this paper, we provide a relationship between equational unification and homological algebra for equational theories. We will construct a functor from the category of sets of equational axioms to the category of abelian groups. Then, our main theorem gives a necessary condition of equational unifiability that is described in terms of abelian groups associated with equational axioms and homomorphisms between them. To construct our functor, we use a ringoid (a category enriched over the category of abelian groups) obtained from the equational axioms and a free resolution of a “good” module over the ringoid, which was developed by Malbos and Mimram.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting

Keywords and phrases Equational unification, Homological algebra, equational theories

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.61

Acknowledgements I would like to thank Assaf Kfoury and Keisuke Nakano for reading a draft of this paper and for their helpful suggestions.

1 Introduction

Equational unification is the problem of solving a given equation modulo an equational theory. For example, if we consider the axioms of commutative rings with a multiplicative unit, the equational unification problem asks whether given a polynomial equation with integer coefficients has a solution in integers. The decidability of this problem was posed by David Hilbert (Hilbert’s tenth problem) and it was shown to be undecidable [12, 5]. There are specific theories such as the theory of abelian groups or the theory of boolean rings such that the equational unification is decidable (see [2, §3.4]). The problem is generally semi-decidable, but not generally decidable. *Narrowing* [6, 7] is a procedure that finds all solutions of the equation, but it may not terminate in general.

Our purpose is to provide a necessary condition of solvability of an equation. The condition is obtained from a homological invariant of equational theories. More precisely, we will define an abelian group $\mathcal{H}(E)$ for a set E of equations (or equational axioms) and an abelian group homomorphism $\mathcal{H}(E \rightarrow E') : \mathcal{H}(E) \rightarrow \mathcal{H}(E')$ for two sets E, E' of equations satisfying $E^* \subset E'^*$. Here, E^*, E'^* are the equational theories of E, E' , i.e., the sets of all equations that can be derived by E, E' . Then, we will prove the following theorem.

► **Theorem 1.** *Let Σ be a signature, E be a set of equations of $\text{Term}(\Sigma)$ and $t, s \in \text{Term}(\Sigma)$ be two terms. If t, s are E -unifiable, then $\mathcal{H}(E \rightarrow E \cup \{t \approx s\})$ is surjective.*

Although $\mathcal{H}(E)$ and $\mathcal{H}(E \rightarrow E')$ are defined using abstract algebra, Theorem 1 is restated in terms of rewriting and matrices if a complete TRS of $E \cup \{t \approx s\}$ is given. In that case, we can compute a matrix associated with E and $E \cup \{t \approx s\}$ and the surjectivity of $\mathcal{H}(E \rightarrow E \cup \{t \approx s\})$ can be checked by matrix operations. (Theorem 5). Therefore, we have



© Mirai Ikebuchi;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 61; pp. 61:1–61:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a sound procedure for non- E -unifiability; if we compute the matrix and if it does not have full rank, then we can conclude that t, s are not E -unifiable. We will see how this procedure works on some simple examples in Section 3.

Our contribution is not only presenting a new procedure for non- E -unifiability. Our abelian group $\mathcal{H}(E)$ and homomorphism $\mathcal{H}(E \rightarrow E')$ can provide an algebraic consideration of equational unification, equational logic, or rewriting. The abelian group $\mathcal{H}(E)$ is invariant under equivalence of E , that is, if two sets E, E' of equations are equivalent, then $\mathcal{H}(E)$ and $\mathcal{H}(E')$ are isomorphic. We define $\mathcal{H}(E)$ using homological algebra of equational theories. The homological algebra of equational theories we use in this paper is based on [10, 11, 8]. Also, we will prove that \mathcal{H} is a functor from the category of sets of equations over a fixed signature to the category of abelian groups.

The paper is organized as follows. In Section 2, we explain basic concepts of unifiability and rewriting. In Section 3, as mentioned earlier, we rephrase Theorem 1 under a certain case so that our condition is checkable by matrix computations. Then, we see some examples and consider equational unification problems on them. In Section 4, we define $\mathcal{H}(E)$ and $\mathcal{H}(E \rightarrow E')$ which appear in Theorem 1 and prove Theorem 1 and 5. In Section 5, we see how homological algebra has been applied to rewriting in other contexts and then we conclude in Section 6.

2 Preliminaries

A *signature* Σ is a set associated with a function $\alpha : \Sigma \rightarrow \mathbb{Z}_{\geq 0}$. For $f \in \Sigma$, we say that f is of *arity* n if $n = \alpha(f)$. Let V be a countably infinite set distinct from Σ . A *term* over Σ and V is a formal object defined inductively as follows:

1. Any element in V , called a *variable*, is a term.
2. For $f \in \Sigma$ of arity n , if t_1, \dots, t_n are terms, then $f(t_1, \dots, t_n)$ is also a term.

Here, $f(t_1, \dots, t_n)$ is a formal expression and not a function application, though its semantics is often treated as a function application. If $c \in \Sigma$ is of arity 0, we write just c for $c()$. For a signature Σ , let $\text{Term}(\Sigma, V)$ denote the set of terms over Σ and V . Also, in this paper, the variables we use are x_1, x_2, \dots , so we just write $\text{Term}(\Sigma)$ for $\text{Term}(\Sigma, \{x_1, x_2, \dots\})$. If f is a symbol of arity 2 that is usually written in infix notation (e.g., $+$, \times), we write $t_1 f t_2$ instead of $f(t_1, t_2)$. We write $\text{Var}(t)$ for the set of variables that occur in t .

A *substitution* is a function $V \rightarrow \text{Term}(\Sigma, V)$. For a term t and a substitution σ , $t\sigma$ denotes the term obtained by replacing all variables v in t with $\sigma(v)$. If a substitution σ satisfies $\sigma(v_1) = t_1, \dots, \sigma(v_n) = t_n$ and $\sigma(v) = v$ for any $v \neq v_1, \dots, v_n$, σ is written as $\{v_1 \mapsto t_1, \dots, v_n \mapsto t_n\}$. Two terms t, s are *unifiable* if there exists a substitution σ such that $t\sigma = s\sigma$. Such σ is called a *unifier*. A *most general unifier (mgu)* of unifiable terms t, s is a unifier σ of t, s satisfying that for any other unifier σ' of t, s , there exists a substitution τ such that $\sigma' = \sigma\tau$.

A *context* is a term in $\text{Term}(\Sigma, V \cup \{\square\})$ that has just one \square in it. For a context $C \in \text{Term}(\Sigma, V \cup \{\square\})$ and a term $t \in \text{Term}(\Sigma, V)$, $C[t]$ denotes the term $C\{\square \mapsto t\}$.

An *equation* is a pair of terms. Equations are written as $l \approx r$. A *rewrite rule* is an equation $l \approx r$ satisfying $\text{Var}(l) \supset \text{Var}(r)$. For rewrite rules, we write $l \rightarrow r$ instead of $l \approx r$. A *term-rewriting system (TRS)* is a set of rewrite rules. For an equation $l \approx r$ and a term t , we say that t is rewritten to s by $l \approx r$, denoted $t \xrightarrow[l \approx r]{} s$, if there is a context

C and a substitution σ such that $t = C[l\sigma]$ and $s = C[r\sigma]$. For a set of equations E and two terms t, s , we say that t is rewritten to s by E , denoted $t \rightarrow_E s$, if $t \xrightarrow[l \approx r]{*} s$ holds for some $l \approx r \in E$. The reflexive transitive closure of the relation \rightarrow_E is written as $\xrightarrow{*}_E$, and the reflexive symmetric transitive closure of \rightarrow_E is written as $\xleftrightarrow{*}_E$ or \approx_E . Two sets E, E' of equations are *equivalent* if $\approx_E = \approx_{E'}$. Two terms t, s are said to be *E -unifiable* if there exists a substitution σ such that $t\sigma \approx_E s\sigma$. Such a σ is called an *E -unifier*. If we consider the problem of finding an E -unifier of two terms t, s , we write $t \approx_E^? s$ for the problem.

A TRS R is *terminating* if there is no infinite path $t_1 \rightarrow_R t_2 \rightarrow_R t_3 \rightarrow_R \dots$.

Two terms t_1, t_2 are *joinable* by R if there exists a term s such that $t_1 \xrightarrow{*}_R s \xleftarrow{*}_R t_2$. A TRS R is *confluent* if, for any terms t, t_1, t_2 , $t_1 \xleftarrow{*}_R t \xrightarrow{*}_R t_2$ implies that t_1 and t_2 are joinable.

A TRS R is *complete* if R is terminating and confluent.

Let R be a TRS and $l_1 \approx r_1, l_2 \approx r_2 \in R$ be two rewrite rules. Suppose that the variables of $l_2 \approx r_2$ are renamed so that $\text{Var}(l_1) \cap \text{Var}(l_2) = \emptyset$. For some context C and nonvariable term t , if t and l_2 are unifiable with mgu σ and if $C[t] = l_1$, then the pair $(r_1\sigma, C[r_2\sigma])$ is called a *critical pair* of R . For example, suppose that we have two rules

$$\begin{aligned} A: & f(f(x_1, x_2), x_3) \approx f(x_1, f(x_2, x_3)) \\ B: & f(i(x_4), x_4) \approx e. \end{aligned}$$

The subterm $f(x_1, x_2)$ of the left-hand side of A and $f(i(x_4), x_4)$, the right-hand side of B , can be unified with the mgu $\sigma = \{x_1 \mapsto i(x_4), x_2 \mapsto x_4, x_4 \mapsto x_4\}$. Then, the corresponding critical pair is $(f(i(x_4), f(x_4, x_3)), e)$, as the following diagram shows.

$$\begin{array}{ccc} & f(f(i(x_4), x_4), x_3) & \\ & \swarrow A \quad \searrow B & \\ f(i(x_4), f(x_4, x_3)) & & e \end{array}$$

3 A Computable Necessary Condition

Let Σ be a signature. For a set E of equations and two terms t, s , if there exists a complete TRS R of $E \cup \{t \approx s\}$, Theorem 1 can be described more explicitly. To state the explicit version of the theorem, we need some definitions.

► **Definition 2.** Let E be a set of equations. The degree of E , denoted by $\text{deg}(E)$, is defined by $\text{deg}(E) = \text{gcd}\{\#_i l - \#_i r \mid l \approx r \in E, i = 1, 2, \dots\}$ where $\#_i t$ is the number of occurrences of x_i in t for $t \in T(\Sigma)$ and $\text{gcd}\{0\}$ is defined to be 0.

For example, $\text{deg}(\{f(x_1, x_2, x_2) \approx x_1, g(x_1, x_1, x_1) \approx e\}) = \text{gcd}\{0, 2, 3\} = 1$.

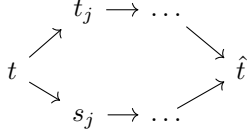
Let $R = \{l_1 \rightarrow r_1, \dots, l_n \rightarrow r_n\}$ be a TRS and $\text{CP}(R) = \{(t_1, s_1), \dots, (t_m, s_m)\}$ be the set of the critical pairs of R . For any $j \in \{1, \dots, m\}$, let a_j^R, b_j^R be the numbers in $\{1, \dots, n\}$ such that the critical pair (t_j, s_j) is obtained by $l_{a_j^R} \rightarrow r_{a_j^R}$ and $l_{b_j^R} \rightarrow r_{b_j^R}$, that is, $t_j = r_{a_j^R}\sigma \leftarrow l_{a_j^R}\sigma = C[l_{b_j^R}\sigma] \rightarrow C[r_{b_j^R}\sigma] = s_j$ for some substitution σ and single-hole context C after suitably renaming variables in $l_{a_j^R} \rightarrow r_{a_j^R}$ and $l_{b_j^R} \rightarrow r_{b_j^R}$. Suppose R is complete. We fix an arbitrary rewriting strategy and for a term t , let $\text{nr}_i^R(t)$ be the number of times $l_i \rightarrow r_i$ is used to reduce t into its R -normal form with respect to the strategy.

61:4 A Homological Condition on Equational Unifiability

For a natural number d , we write \mathbb{Z}_d for $\mathbb{Z}/d\mathbb{Z}$, the integers modulo d .

► **Definition 3.** Let $d = \deg(R)$. The matrix $D(R)$ is a $n \times m$ matrix over \mathbb{Z}_d whose (i, j) -th entry $D(R)_{ij}$ ($1 \leq i \leq n$, $1 \leq j \leq m$) is $[\text{nr}_i^R(s_j) - \text{nr}_i^R(t_j) + \delta(b_j^R, i) - \delta(a_j^R, i)] \in \mathbb{Z}_d$ where $\delta(x, y)$ is the Kronecker delta. (That is, $\delta(x, y) = 1$ if $x = y$ and 0 if $x \neq y$.)

In other words, the (i, j) -th entry of $D(R)$ is the difference between (1) the number of $l_i \rightarrow r_i$ in the upper path from t to \hat{t} in the diagram below, and (2) that in the lower path.



The degree $\deg(E)$ and the matrix $D(R)$ are introduced in [8] to give a lower bound of number of equational axioms that is needed to present a given equational theory.

► **Definition 4.** Let $E = \{l'_1 \approx r'_1, \dots, l'_{n'} \approx r'_{n'}\}$ be a set of equations and $R = \{l_1 \rightarrow r_1, \dots, l_n \rightarrow r_n\}$ be a complete TRS. Suppose $E^* \subset R^*$. Then, $U(E, R)$ is the $n \times n'$ matrix over \mathbb{Z}_d whose (i, j) -th entry is $[\text{nr}_i^R(l'_j) - \text{nr}_i^R(r'_j)] \in \mathbb{Z}_d$ where $d = \deg(R)$.

For a commutative ring A , two $n \times m$ matrices M, N over A are said to be *equivalent* if $N = PMQ$ for some invertible $n \times n$ matrix P and $m \times m$ matrix Q over A . We write $I_{n,m}$ for the $n \times m$ diagonal matrix whose diagonal elements are all 1.

Here is the explicit version of Theorem 1.

► **Theorem 5.** Let $E = \{u_1 \approx v_1, \dots, u_k \approx v_k\}$ be a set of equations and t, s be two terms. Suppose that there is a complete TRS $R = \{l_1 \rightarrow r_1, \dots, l_n \rightarrow r_n\}$ of $E \cup \{t \approx s\}$ and $\deg(R) \neq 1$. If t, s are E -unifiable, then the augmented matrix $(D(R)|U(E, R))$ is equivalent to $I_{n,m}$ and $n \leq m$ where m is the number of columns of $(D(R)|U(E, R))$.

We will prove Theorem 1 and how it implies Theorem 5 in Section 4 after introducing more algebraic tools.

► **Remark 6.** Although the matrices $D(R)$ and $U(E, R)$ depend on the choice of rewriting strategy, the necessary condition stated in Theorem 5 does not depend on the choice. We will prove this fact in Section 4.

► **Remark 7.** It is algorithmically checkable whether a matrix over \mathbb{Z}_d is equivalent to $I_{n,m}$ in polynomial time by computing the Smith normal form [4, Chapter 15]. Note that if the degree d is prime, since \mathbb{Z}_d is a field, it suffices to get a diagonal matrix by elementary row/column operations and see all diagonal elements are nonzero.

We shall see some examples.

► **Example 8.** Let E_1 be the set of equations

$$B_1 : 0 + x_1 \approx x_1, \quad B_2 : s(x_1) + x_2 \approx s(x_1 + x_2).$$

Consider the E_1 -unification problem $x_1 + x_1 \stackrel{?}{\approx}_{E_2} s(0)$. By applying Knuth-Bendix completion to $E_1 \cup \{x_1 + x_1 \rightarrow s(0)\}$, we obtain a complete TRS R_1 :

$$B_1 : 0 + x_1 \rightarrow x_1, \quad C_1 : x_1 + x_1 \rightarrow 0, \quad C_2 : s(x_1) \rightarrow x_1.$$

The degree of R_1 is 2 and R_1 has one critical pair $\Pi' : 0 \xleftarrow{B_1} 0 + 0 \xrightarrow{C_1} 0$ and the matrix $(D(R_1)|U(E_1, R_1))$ is given as

$$\begin{array}{c|cc} \Pi' & B_1 & B_2 \\ \hline B_1 & 1 & 0 \\ C_1 & 1 & 0 \\ C_2 & 0 & 0 \end{array}.$$

Here, each entry of $D(R_1|U(E_1, R_1))$ is thought of as an element in \mathbb{Z}_2 . Since it does not have full rank, $x_1 + x_1$ and $s(0)$ are not E_1 -unifiable.

We can also consider the E_1 -unification problem $x_1 + x_1 \approx_{E_2}^? 0$. It has an obvious solution $x_1 \mapsto 0$, so the matrix corresponding to this problem must be equivalent to $I_{n,m}$ for some n, m . We give a complete TRS for $E_1 \cup \{x_1 + x_1\}$, its critical pairs, and the corresponding matrix in the appendix.

More generally, consider the E_1 -unification problem $x_1 + x_1 \approx_{E_1}^? s^n(0)$ where $s^n(0) = s(\dots s(0)\dots)$. In fact, we can see that if n is odd, $E' = E_1 \cup \{x_1 + x_1 \approx s^n(0)\}$ is equivalent to $\underbrace{\overline{E_1}}_n \cup \{x_1 + x_1 \approx s(0)\}$ and if n is even, E' is equivalent to $E_1 \cup \{x_1 + x_1 \approx 0\}$.

► **Example 9.** Let $E_2 = \{a(b(b(a(x_1)))) \approx x_1\}$. It is known that E_2 does not have a complete TRS with a finite number of rewrite rules [9]. Consider the E_2 -unification problem $a(b(x_1)) \approx_{E_2}^? x_1$. Then, $E_2 \cup \{a(b(x_1)) \approx x_1\}$ has a complete TRS $R_2 = \{a(b(x_1)) \rightarrow x_1, b(a(x_1)) \rightarrow x_1\}$. Then, there are two critical pairs

$$\begin{array}{l} a(x_1) \xleftarrow{a(b(x_1)) \rightarrow x_1} a(b(a(x_1))) \xrightarrow{b(a(x_1)) \rightarrow x_1} a(x_1), \\ b(x_1) \xleftarrow{a(b(x_1)) \rightarrow x_1} b(a(b(x_1))) \xrightarrow{b(a(x_1)) \rightarrow x_1} b(x_1). \end{array}$$

It is easy to check that $(D(R_2)|U(E_2, R_2))$ is the 2×3 matrix whose entries are all 1 and so it is not equivalent to $I_{2,3}$. Therefore, $a(b(x_1))$ and x_1 are not E_2 -unifiable.

► **Remark 10.** As Example 9 indicates, it can be the case that it is difficult or impossible to find a complete TRS of the given set E of equations but a complete TRS of $E \cup \{t \approx s\}$ is easy to find. The basic version of narrowing, the main existing tool for E -unification for unspecified E , is applicable only when a complete TRS of E is given. So, it is notable that Theorem 5 does not require us to find a complete TRS of E .

4 Homological Algebra on Equational Theories

The aim of this section is to define $\mathcal{H}(E)$ and $\mathcal{H}(E \rightarrow E')$, and to prove Theorem 1 and Theorem 5. For that, we will construct some algebraic structures associated with E and applies homological algebra to them. First, let us see the notion of *resolution*, which is often used to define invariants of mathematical objects in many branches of mathematics. See [15] as an introductory text.

Let R be a ring. We say that the sequence

$$\dots \xrightarrow{f_{i+1}} M_{i+1} \xrightarrow{f_i} M_i \xrightarrow{f_{i-1}} M_{i-1} \xrightarrow{f_{i-2}} \dots$$

of left R -modules M_i and R -linear maps f_i is *exact* if $\ker f_i = \text{im } f_{i+1}$ holds. For a left R -module M , a *free resolution* of M is an exact sequence

$$\dots \xrightarrow{\partial_2} F_2 \xrightarrow{\partial_1} F_1 \xrightarrow{\partial_0} F_0 \xrightarrow{\epsilon} M \rightarrow 0$$

61:6 A Homological Condition on Equational Unifiability

where each F_i is free. It is known that for any left R -module M , free resolutions of M exist. A *partial free resolution* of M is an exact sequence of finite length

$$F_n \xrightarrow{\partial_n} \dots \xrightarrow{\partial_2} F_2 \xrightarrow{\partial_1} F_1 \xrightarrow{\partial_0} F_0 \xrightarrow{\epsilon} M \rightarrow 0$$

with free F_i s.

The notion of resolution is defined not only for modules over a ring but also for modules over a *ringoid*. In [11], Malbos and Mimram constructed a ringoid associated with a given equational theory and defined invariants called homology groups using a free resolution over that ringoid. We will also use the free resolution to define $\mathcal{H}(E)$ in Theorem 1. We shall see their construction in the subsections from 4.1 to 4.3, then provide the definitions of $\mathcal{H}(E)$, $\mathcal{H}(E \rightarrow E')$ and prove our main theorems.

4.1 Category of Bicontexts

We fix a signature Σ . Let $t = \langle t_1, \dots, t_n \rangle$ be an n -tuple of terms whose variables are in $\{x_1, \dots, x_m\}$ and $s = \langle s_1, \dots, s_m \rangle$ be an m -tuple of terms. We define their composition $t \circ s$ by $\langle t_1[s_1/x_1, \dots, s_m/x_m], \dots, t_n[s_1/x_1, \dots, s_m/x_m] \rangle$ where $t_i[s_1/x_1, \dots, s_m/x_m]$ is the term obtained by substituting s_j for x_j in t_i for each $j = 1, \dots, m$ in parallel.

► **Definition 11.** A bicontext is a pair (C, t) of a context C and n -tuple of terms $t = \langle t_1, \dots, t_n \rangle$.

For two bicontexts (C, t) and (D, s) , we define their composition $(C, t) \circ (D, s)$ by $(C[D \circ t], s \circ t)$ where $D \circ t = D[t_1/x_1, \dots, t_n/x_n]$ for $t = \langle t_1, \dots, t_n \rangle$.

► **Definition 12.** The category of bicontexts \mathbb{K} consists of

- *Objects:* natural numbers $0, 1, \dots$,
- *Morphisms* $\mathbb{K}(n_1, n_2)$: bicontexts (C, t) where t is an n_1 -tuple of terms such that the elements of t and C have variables in $\{x_1, \dots, x_{n_2}\}$ (except \square in C),
- *Identity:* $(\square, \langle x_1, \dots, x_n \rangle)$

and the composition is defined above.

4.2 Ringoids

We consider an algebraic structure called *ringoid*.

► **Definition 13.** A ringoid \mathcal{R} is a small **Ab**-enriched category. That is, each hom-set is equipped with abelian group structure $(\text{hom}_{\mathcal{R}}(a, b), +, 0)$ and satisfies the following rules.

$$0 \circ x = 0, \quad x \circ 0 = 0, \quad z \circ (x + y) = z \circ x + z \circ y, \quad (z + w) \circ x = z \circ x + w \circ x$$

where $x, y \in \text{hom}_{\mathcal{R}}(a, b)$, $z, w \in \text{hom}_{\mathcal{R}}(b, c)$.

A ringoid can be thought of as a “many-sorted” ring. If a ringoid has just a single object, its morphisms form a ring with addition $+$ and multiplication \circ . If a ringoid has multiple objects, each object can be thought of as a sort. We can add two morphisms $x : a_1 \rightarrow b_1$, $y : a_2 \rightarrow b_2$ only if $a_1 = a_2$ and $b_1 = b_2$. Also, we can multiply them as composition $y \circ x$ only if $b_1 = a_2$.

For any small category \mathcal{C} , there exists a ringoid $\mathbb{Z}\langle\mathcal{C}\rangle$ called the ringoid *freely generated by* \mathcal{C} . The ringoid $\mathbb{Z}\langle\mathcal{C}\rangle$ has the same objects as \mathcal{C} and the hom-set $\mathbb{Z}\langle\mathcal{C}\rangle(a, b)$ between objects a, b is the free abelian group generated by $\mathcal{C}(a, b)$. The composition of $\mathbb{Z}\langle\mathcal{C}\rangle$ is given by linearly extending the composition of \mathcal{C} as $(w + z) \circ (x + y) = w \circ x + w \circ y + z \circ x + z \circ y$.

We can define an ideal of a ringoid and a module over a ringoid.

► **Definition 14.** Let \mathcal{R} be a ringoid. An ideal of \mathcal{R} is a subfunctor of the hom-bifunctor $\mathcal{R}(-, -) : \mathcal{R} \times \mathcal{R} \rightarrow \mathbf{Ab}$. If I is an ideal of \mathcal{R} , then we can define the category \mathcal{R}/I whose objects are those of \mathcal{R} , morphisms are $(\mathcal{R}/I)(a, b) = \mathcal{R}(a, b)/I(a, b)$, and the composition is induced by that of \mathcal{R} . Also, a structure of ringoid of \mathcal{R}/I is induced by that of \mathcal{R} .

► **Definition 15.** Let \mathcal{R} be a ringoid.

- A left \mathcal{R} -module is a functor $M : \mathcal{R} \rightarrow \mathbf{Ab}$ satisfying $M(x+y) = M(x) + M(y)$, $M(0) = 0$ for any $x, y \in \mathcal{R}(a, b)$, $a, b \in \text{Obj}(\mathcal{R})$. We define the scalar multiplication $\cdot : \mathcal{R}(a, b) \times M(a) \rightarrow M(b)$ as $a \cdot m = M(a)(m)$.
- A right \mathcal{R} -module is a left \mathcal{R}^{op} -module.
- For two left \mathcal{R} -modules M_1, M_2 , an \mathcal{R} -linear map $f : M_1 \rightarrow M_2$ is a natural transformation. (We can define an \mathcal{R} -linear map between right \mathcal{R} -modules in the same manner.)

► **Definition 16.** Let M_1 be a left \mathcal{R} -module. A submodule of M_1 is a left \mathcal{R} -module M_2 such that there exists a monomorphism $\phi : M_2 \rightarrow M_1$ and $\phi_a : M_2(a) \rightarrow M_1(a)$ is an inclusion of sets for each object a of \mathcal{R} .

We define left free \mathcal{R} -modules over ringoids.

► **Definition 17.** Let P be a family of sets P_a ($a \in \text{Obj}(\mathcal{R})$). The left free \mathcal{R} -module generated by P , denoted by $\mathcal{R}\underline{P}$, is defined as follows. For each $a \in \text{Obj}(\mathcal{R})$, $(\mathcal{R}\underline{P})(a)$ is the abelian group consisting of formal finite sums $\sum_{x \in P_b, b \in \text{Obj}(\mathcal{R})} \lambda_x \underline{x}$, ($\lambda_x \in \mathcal{R}(b, a)$). Here, the underline for x above is added to emphasize the difference between λ_x and x . The scalar multiplication is given as $r \cdot (\sum_x \lambda_x \underline{x}) = \sum_x (r \circ \lambda_x) \underline{x}$ ($r \in \mathcal{R}(a, c)$).

For a ringoid \mathcal{R} , let $\mathbf{Mod}_{\mathcal{R}}$ denote the category of left \mathcal{R} -modules and \mathcal{R} -linear maps. The following proposition tells us that $\mathbf{Mod}_{\mathcal{R}}$ has good properties so that we can apply homological algebra to it.

► **Proposition 18** ([13]). $\mathbf{Mod}_{\mathcal{R}}$ is an abelian category and any left \mathcal{R} -module has a free resolution.

We do not give the details of this proposition, but one of the important consequences of being abelian is that we have the notions of kernel and image of an \mathcal{R} -linear map in the category.

► **Definition 19.** Let M_1, M_2 be two left \mathcal{R} -modules and $f : M_1 \rightarrow M_2$ be an \mathcal{R} -linear map. Then, the kernel and the image of f are defined as $(\ker f)(a) = \ker f_a$, $(\text{im } f)(a) = \text{im } f_a$ for each object a . Here, f_a is an abelian group homomorphism, so \ker and im in the right-hand sides are the kernel and the image for group homomorphisms.

Many other notions for modules over a ring can be generalized.

► **Definition 20.** Let M_1 be a left \mathcal{R} -module and M_2 be a submodule of M_1 . The quotient module M_1/M_2 is the left \mathcal{R} -module given as $(M_1/M_2)(a) = M_1(a)/M_2(a)$.

► **Definition 21.** Let M be a left \mathcal{R} -module. For an index set I , for each $i \in I$, let a_i be an object of \mathcal{R} and x_i be an element of $M(a_i)$. The submodule generated by $\{x_i\}_{i \in I}$ is the left \mathcal{R} -module N such that for each $a \in \text{Obj}(\mathcal{R})$, $N(a)$ is the abelian group consisting of finite sums $\sum_{i \in I} \lambda_i \cdot x_i$ ($\lambda_i \in \mathcal{R}(a, a_i)$).

► **Definition 22.** Let M_1 be a right \mathcal{R} -module and M_2 be a left \mathcal{R} -module. The tensor product $M_1 \otimes_{\mathcal{R}} M_2$ of M_1 and M_2 is defined as the coend $M_1 \otimes_{\mathcal{R}} M_2 = \int^a M_1(a) \otimes M_2(a)$. That is, an abelian group $M_1 \otimes_{\mathcal{R}} M_2$ is the tensor product of M_1, M_2 if there is an extranatural transformation $\zeta : M_1(-) \otimes M_2(-) \rightarrow M_1 \otimes_{\mathcal{R}} M_2$ such that for any abelian group A and any extranatural transformation $\gamma : M_1(-) \otimes M_2(-) \rightarrow A$, there exists a unique abelian group homomorphism $\phi : M_1 \otimes_{\mathcal{R}} M_2 \rightarrow A$ with $\gamma_a = \phi \circ \zeta_a$ for any $a \in \text{Obj}(\mathcal{R})$.

Explicitly, $M_1 \otimes_{\mathcal{R}} M_2$ is the abelian group $\left(\bigoplus_{a \in \text{Obj}(\mathcal{R})} M_1(a) \otimes M_2(a) \right) / R$ where R is the abelian group generated by $M_1(f \circ p)(x) \otimes y - x \otimes M_2(f)(y)$ for any $f : a \rightarrow a', x \in M_1(a'), y \in M_2(a), a, a' \in \text{Obj}(\mathcal{R})$.

Let M_1, M_2 be two left \mathcal{R} -modules and N_1, N_2 be two right \mathcal{R} -modules. For linear maps $f : M_1 \rightarrow M_2$ and $g : N_1 \rightarrow N_2$, we define $g \otimes f : N_1 \otimes_{\mathcal{R}} M_1 \rightarrow N_2 \otimes_{\mathcal{R}} M_2$ to be the abelian group homomorphism $(f \otimes g)(n \otimes m) = g(n) \otimes f(m)$. If $N_1 = N_2$ and g is the identity map, we write $N_1 \otimes f$ instead of $g \otimes f$.

The tensor product of modules over a ringoid satisfies many properties of tensor product of modules over a ring. In particular, we have

► **Lemma 23.** Let N be a right \mathcal{R} -module and M_1, M_2, M_3 be left \mathcal{R} -modules. If the sequence $M_1 \xrightarrow{f} M_2 \xrightarrow{g} M_3 \rightarrow 0$ is exact, then the sequence $N \otimes_{\mathcal{R}} M_1 \xrightarrow{N \otimes f} N \otimes_{\mathcal{R}} M_2 \xrightarrow{N \otimes g} N \otimes_{\mathcal{R}} M_3 \rightarrow 0$ is also exact.

4.3 Partial Free Resolutions

For the category of bicontexts \mathbb{K} , consider the ringoid $\mathbb{Z}\langle\mathbb{K}\rangle$. Then, we will define a ringoid \mathcal{R}^E such that any two equivalent sets E, E' of equations give rise to isomorphic ringoids $\mathcal{R}^E \simeq \mathcal{R}^{E'}$.

For a term t and a positive integer i , let $\kappa_i(t)$ be the linear combination of contexts given inductively by

$$\kappa_i(x_i) = \square, \quad \kappa_i(x_j) = 0 \quad (i \neq j), \quad \kappa_i(f(t_1, \dots, t_k)) = \sum_{j=1}^k f(t_1, \dots, \underbrace{\square}_{j\text{th}}, \dots, t_k) [\kappa_i(t_j)].$$

Application of linear combination of contexts to a context which appears in the last rule is defined by $C[D_1 + \dots + D_n] = C[D_1] + \dots + C[D_n]$. Also, for a term t , symbol $f \in \Sigma^{(n)}$, and n -uple of terms $u = \langle s_1, \dots, s_n \rangle$, let $\varphi_{f,u}(t)$ be the linear combination of all contexts C satisfying $C[f(s_1, \dots, s_n)] = t$.

We define the ideal I_E of $\mathbb{Z}\langle\mathbb{K}\rangle$. Let $I_E(m, n)$ be the subgroup of $\mathbb{Z}\langle\mathbb{K}\rangle(m, n)$ generated by elements of the form

$$(\kappa_i(s) - \kappa_i(t), w), \quad (\varphi_{f,vu}(t \circ v) - \varphi_{f,vu}(s \circ v) - \varphi_{f,u}(t) \circ v + \varphi_{f,u}(s) \circ v, w), \quad (\square, w_1) - (\square, w_2)$$

for any $s \approx_E t, w_1 \approx_E w_2$. Then, define \mathcal{R}^E to be $\mathbb{Z}\langle\mathbb{K}\rangle / I_E$.¹ For a morphism x of $\mathbb{Z}\langle\mathbb{K}\rangle$, we write $[x]^E$ or just $[x]$ for the equivalence class of x in \mathcal{R}^E . If we consider the free module $\mathcal{R}^E \underline{P}$ for a family P of sets P_0, P_1, \dots , we write $C_1 \underline{p}u_1 + \dots + C_k \underline{p}u_k$ for $[(C_1, u_1) + \dots + (C_k, u_k)] \underline{p} \in \mathcal{R}^E \underline{P}(i)$. By definition, for any E' equivalent to E , $\mathcal{R}^{E'}$ is isomorphic to \mathcal{R}^E .

¹ For the original definition of \mathcal{R}^E in [11], the generators $\varphi_{f,vu}(t \circ v) - \varphi_{f,vu}(s \circ v) - \varphi_{f,u}(t) \circ v + \varphi_{f,u}(s) \circ v$ of $I_E(m, n)$ was not given. However, we need these generators to prove $\partial_1(\hat{t}) = \varphi(\hat{t}) - \varphi(t)$ which is used to show $\partial_1 \circ \partial_2 = 0$ in Appendix A of [11]. We do not need to change the other parts of the proof.

Let $d = \deg(E)$. Consider the right \mathcal{R}^E -module that maps any object n to \mathbb{Z}_d and whose scalar multiplication $\cdot : \mathbb{Z}_d \times \mathcal{R}^E(m, n) \rightarrow \mathbb{Z}_d$ is given by $[1] \cdot [(C_1, t_1) + \dots + (C_k, t_k)] = [k]$. We write \mathbb{Z}_d also for this right \mathcal{R}^E -module. We show that the scalar multiplication is well-defined. If $C_1 + \dots + C_k = \kappa_i(s)$ and $D_1 + \dots + D_{k'} = \kappa_i(t)$ for some $s \approx t$, then $k - k' = \#_i s - \#_i t$ is divided by d by the definition of $\deg(E)$. Thus, $[1] \cdot [\sum_{i=1}^k (C_i, t) - \sum_{i=1}^{k'} (D_i, t)] = [0]$. Also, since the number of bicontexts in $\varphi_{f,u}(t)$ is the number of subterm $f(u)$ in t , for any $l \approx r \in E$, $f \in \Sigma$, $t \in T(\Sigma)$, the linear combination $\varphi_{f,u}(r \circ t) - \varphi_{f,u}(l \circ t) - \varphi_{f,u}(r) + \varphi_{f,u}(l)$ consists of da contexts for some nonnegative integer a . Therefore, $[1] \cdot [\varphi_{f,u}(r \circ t) - \varphi_{f,u}(l \circ t) - \varphi_{f,u}(r) + \varphi_{f,u}(l)] = [0]$, so the scalar multiplication for \mathbb{Z}_d is well-defined.

Let X_1 be a singleton set $\{\star\}$, X_i be the empty set for $i = 0$ or $i = 2, 3, \dots$, and X be the family consisting of X_i s. We define a left \mathcal{R}^E -module \mathcal{Z}^E to be the quotient $\mathcal{R}^E \underline{X} / N$ where N is the submodule of $\mathcal{R}^E \underline{X}$ generated by $\sum_{i=1}^m \kappa_i(u) \circ t \star t_i - \square_{\star} \langle u \circ t \rangle$ for every term u with $\text{Var}(u) \subset \{x_1, \dots, x_m\}$ and m -uple $t = \langle t_1, \dots, t_m \rangle$ of terms. Then, we construct a partial free resolution of \mathcal{Z}^E

$$\mathcal{R}^E \underline{\mathbf{P}}_2 \xrightarrow{\partial_2^E} \mathcal{R}^E \underline{\mathbf{P}}_1 \xrightarrow{\partial_1^E} \mathcal{R}^E \underline{\mathbf{P}}_0 \xrightarrow{\epsilon^E} \mathcal{Z}^E \rightarrow 0 \quad (1)$$

as follows. First, $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2^E$ are families of sets $(\mathbf{P}_0)_j, (\mathbf{P}_1)_j, (\mathbf{P}_2^E)_j$ given as

$$(\mathbf{P}_0)_j = \begin{cases} \{1\} & (j = 1) \\ \emptyset & (j \neq 1) \end{cases}, \quad (\mathbf{P}_1)_j = \Sigma^{(j)} = \{f \in \Sigma \mid f \text{ has arity } j\}$$

$$(\mathbf{P}_2^E)_j = \{l \approx r \in E \mid \text{Var}(l) \cup \text{Var}(r) \subset \{x_1, \dots, x_j\}\}.$$

Then, we define \mathcal{R}^E -linear maps $\epsilon^E, \partial_0^E, \partial_1^E$ as

$$\epsilon^E(\underline{1}) = \underline{\star}, \quad \partial_0^E(f) = \sum_{i=1}^n f(x_1, \dots, \underbrace{\square}_{i\text{th}}, \dots, x_n) \underline{1}\langle x_i \rangle - \underline{1}\langle f(x_1, \dots, x_n) \rangle,$$

$$\partial_1^E(l \approx r) = \varphi(r) - \varphi(l)$$

where $\varphi : \text{Term}(\Sigma) \rightarrow \mathcal{R}^E \underline{\mathbf{P}}_1$ is defined inductively as

$$\varphi(x_i) = 0, \quad \varphi(f(t_1, \dots, t_n)) = \underline{f}\langle t_1, \dots, t_n \rangle + \sum_{i=1}^n \underline{f}\langle t_1, \dots, \underbrace{\square}_{i\text{th}}, \dots, t_n \rangle \varphi(t_i).$$

If there is a complete TRS R of E , we can extend the sequence (1) to

$$\mathcal{R}^R \underline{\mathbf{P}}_3 \xrightarrow{\partial_3^R} \mathcal{R}^R \underline{\mathbf{P}}_2 \xrightarrow{\partial_2^R} \mathcal{R}^R \underline{\mathbf{P}}_1 \xrightarrow{\partial_1^R} \mathcal{R}^R \underline{\mathbf{P}}_0 \xrightarrow{\epsilon^R} \mathcal{Z}^R \rightarrow 0. \quad (2)$$

Here, \mathbf{P}_3^R is the family of sets $(\mathbf{P}_3^R)_j$ where each $(\mathbf{P}_3^R)_j$ consists of 5-uple $(l \rightarrow r, t, C, l' \rightarrow r', t')$ such that

- $l \circ t = C[l' \circ t']$ and $r \circ t \leftarrow l \circ t = C[l' \circ t'] \rightarrow C[r' \circ t']$ is a critical peak, and
- either $l \rightarrow r$ or $l' \rightarrow r'$ is in $(\mathbf{P}_2^R)_j$ and the other is in $(\mathbf{P}_2^R)_k$ for some $k \leq j$.

For such a 5-uple $\alpha = (l \rightarrow r, t, C, l' \rightarrow r', t')$, $\partial_2^R(\underline{\alpha})$ is defined as

$$\partial_2^R(\underline{\alpha}) = \underline{l' \rightarrow r' t'} - \underline{C l \rightarrow r t} + \widehat{\underline{r' \circ t'}} - \widehat{\underline{C[r \circ t]}}$$

where \hat{s} is defined for any term s as follows. Suppose s is rewritten to its normal form \hat{s} by rewrite rules $p_1 \rightarrow q_1, \dots, p_k \rightarrow q_k \in R$ as

$$s = C_1[p_1 \circ u_1], C_1[q_1 \circ u_1] = C_2[p_2 \circ u_2], \dots, C_{k-1}[q_{k-1} \circ u_{k-1}] = C_k[p_k \circ u_k], C_k[q_k \circ u_k] = \hat{s}$$

for some C_i s and u_i s. Then, $\hat{s} = \sum_{i=1}^k C_i \underline{p_i} \rightarrow q_i \underline{u_i}$.

61:10 A Homological Condition on Equational Unifiability

► **Theorem 24** ([11]). *If R is a complete TRS, the sequence (2) is exact.*

The following lemma is useful for the next subsection.

► **Lemma 25.** *Let E be a set of equations with degree d . For any family P of sets P_0, P_1, \dots , we have an abelian group isomorphism $\mathbb{Z}_d \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{P} \simeq \mathbb{Z}_d \underline{\uplus} P$ where $\underline{\uplus} P$ is the disjoint union of P_i s and the right-hand side is the free module generated by $\underline{\uplus} P$ over \mathbb{Z}_d as a ring.*

Proof. Consider the abelian group homomorphism $\psi : \mathbb{Z}_d \underline{\uplus} P \rightarrow \mathbb{Z}_d \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{P}$, $p \mapsto 1 \otimes p$. Then, ψ is surjective since $1 \otimes Cpu = 1 \cdot [(C, u)] \otimes p = 1 \otimes p$ for any $1 \otimes Cpu \in \mathbb{Z}_d \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{P}$. Let $\gamma_i : \mathbb{Z}_d \otimes (\mathcal{R}^E \underline{P}(i)) \rightarrow \mathbb{Z}_d \underline{\uplus} P$ be the abelian group homomorphism $1 \otimes Cpu \mapsto p$. We can check that γ_i s form an extranatural transformation γ , so we have $\phi : \mathbb{Z}_d \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{P} \rightarrow \mathbb{Z}_d \underline{\uplus} P$ with $\gamma_i = \phi \circ \zeta_i$ for $\zeta_i : \mathbb{Z}_d \otimes (\mathcal{R}^E \underline{P}(i)) \rightarrow \mathbb{Z}_d \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{P}$. Then, $\phi(\psi(p)) = \phi(\zeta_i(1 \otimes p)) = \gamma_i(1 \otimes p) = p$. Thus, ψ is an isomorphism. ◀

4.4 Invariant $\mathcal{H}(E)$

We are ready to define $\mathcal{H}(E)$.

► **Definition 26.** *For a set E of equations, we define the abelian group $\mathcal{H}(E)$ by*

$$\mathcal{H}(E) = \mathbb{Z}_d \otimes_{\mathcal{R}^E} \ker \partial_0^E = \mathbb{Z}_d \otimes_{\mathcal{R}^E} \text{im } \partial_1^E \quad (d = \text{deg}(E)).$$

If two sets E, E' of equations are equivalent, since \mathcal{R}^E and $\mathcal{R}^{E'}$ are isomorphic and $\partial_0^E = \partial_0^{E'}$, we have $\mathcal{H}(E) \simeq \mathcal{H}(E')$. That is, we can see that $\mathcal{H}(E)$ is invariant under the equivalence of E . (This holds especially since we are fixing a signature Σ .)

Let E, E' be sets of equations with $E^* \subset E'^*$. Then, the functor $\pi^{E, E'} : \mathcal{R}^E \rightarrow \mathcal{R}^{E'}$ given as $[(C_1, u_1) + \dots + (C_k, u_k)]^E \mapsto [(C_1, u_1) + \dots + (C_k, u_k)]^{E'}$ is well-defined. For a family of sets P , $\pi^{E, E'}$ extends to $\bar{\pi}_P^{E, E'} : \mathcal{R}^E \underline{P} \rightarrow \mathcal{R}^{E'} \underline{P}$. Then, we can see that the diagram

$$\begin{array}{ccc} \mathcal{R}^E \underline{\mathbf{P}}_1 & \xrightarrow{\partial_0^E} & \mathcal{R}^E \underline{\mathbf{P}}_0 \\ \downarrow \bar{\pi}_{\underline{\mathbf{P}}_1}^{E, E'} & & \downarrow \bar{\pi}_{\underline{\mathbf{P}}_0}^{E, E'} \\ \mathcal{R}^{E'} \underline{\mathbf{P}}_1 & \xrightarrow{\partial_0^{E'}} & \mathcal{R}^{E'} \underline{\mathbf{P}}_0 \end{array}$$

commutes. Therefore, if we restrict $\bar{\pi}_{\underline{\mathbf{P}}_1}^{E, E'}$ to $\ker \partial_0^E$, we get $\bar{\pi}_{\underline{\mathbf{P}}_1}^{E, E'}|_{\ker \partial_0^E} : \ker \partial_0^E \rightarrow \ker \partial_0^{E'}$. Let $d = \text{deg}(E)$ and $d' = \text{deg}(E')$. Since $E^* \subset E'^*$, d' divides d and we can define a group homomorphism $q^{d, d'} : \mathbb{Z}_d \rightarrow \mathbb{Z}_{d'}$ as $q^{d, d'}(n + d\mathbb{Z}) = n + d'\mathbb{Z}$. Consider the composition of abelian group homomorphisms

$$\mathbb{Z}_d \otimes (\ker \partial_0^E(k)) \xrightarrow{f_k} \mathbb{Z}_{d'} \otimes (\ker \partial_0^{E'}(k)) \xrightarrow{\zeta_k} \mathbb{Z}_{d'} \otimes_{\mathcal{R}^{E'}} \ker \partial_0^{E'}$$

where $f_k = q^{d, d'} \otimes (\bar{\pi}_{\underline{\mathbf{P}}_1}^{E, E'}|_{\ker \partial_0^E(k)})$ and ζ_k is the extranatural transformation given in the definition of tensor product. Since $\zeta_k \circ f_k$ ($k = 0, 1, \dots$) form an extranatural transformation, we get an abelian group homomorphism $\mathbb{Z}_d \otimes_{\mathcal{R}^E} \ker \partial_0^E \rightarrow \mathbb{Z}_{d'} \otimes_{\mathcal{R}^{E'}} \ker \partial_0^{E'}$ by naturality and let $\mathcal{H}(E \rightarrow E')$ denote it. That is, $\mathcal{H}(E \rightarrow E')$ makes the following diagram commute.

$$\begin{array}{ccc} \mathbb{Z}_d \otimes (\ker \partial_0^E(k)) & \xrightarrow{\zeta_k} & \mathbb{Z}_d \otimes_{\mathcal{R}^E} \ker \partial_0^E \\ \downarrow f_k & & \downarrow \mathcal{H}(E \rightarrow E') \\ \mathbb{Z}_{d'} \otimes (\ker \partial_0^{E'}(k)) & \xrightarrow{\zeta_k} & \mathbb{Z}_d \otimes_{\mathcal{R}^{E'}} \ker \partial_0^{E'} \end{array} \quad (3)$$

Thus, we have obtained an abelian group homomorphism $\mathcal{H}(E \rightarrow E') : \mathcal{H}(E) \rightarrow \mathcal{H}(E')$.

Now, we can prove Theorem 1.

Proof of Theorem 1. Let $F = E \cup \{t \approx s\}$. If $t\sigma \approx_E s\sigma$ for some σ , then E is equivalent to $E' = E \cup \{t\sigma \approx s\sigma\}$ and F is equivalent to $F' = F \cup \{t\sigma \approx s\sigma\}$. Since $\mathbb{Z}_d \otimes_{\mathcal{R}^{F'}} \mathcal{R}^{F'} \underline{\mathbf{P}}_2^{F'}$ is freely generated by $1 \otimes \underline{l \approx r}$ for $l \approx r \in F'$ (Lemma 25), $\mathcal{H}(F') = \mathbb{Z}_d \otimes_{\mathcal{R}^{F'}} \text{im } \partial_1^{F'}$ is generated by $1 \otimes \partial_1^{F'}(\underline{l \approx r})$ for $l \approx r \in F'$. For $l \approx r \in E'$, since $\mathcal{H}(E' \rightarrow F')(1 \otimes \partial_1^{E'}(\underline{l \approx r})) = 1 \otimes \partial_1^{F'}(\underline{l \approx r})$, to show the surjectivity of $\mathcal{H}(E' \rightarrow F')$, it suffices to check that $1 \otimes \partial_1^{F'}(\underline{t \approx s})$ is in $\text{im } \mathcal{H}(E' \rightarrow F')$. We have $1 \otimes \partial_1^{F'}(\underline{t \approx s} - \underline{t\sigma \approx s\sigma}) = 0 \in \mathbb{Z}_d \otimes \text{im } \partial_1^{F'}$ since

$$\begin{aligned} 1 \otimes \partial_1^{F'}(\underline{t \approx s} - \underline{t\sigma \approx s\sigma}) &= 1 \otimes (\varphi(s) - \varphi(t) - \varphi(s\sigma) + \varphi(t\sigma)) \\ &= 1 \otimes (\varphi(s) - \varphi(t)) - 1 \otimes (\varphi(s\sigma) - \varphi(t\sigma)) \\ &= 1 \otimes (\varphi(s)\sigma - \varphi(t)\sigma) - 1 \otimes (\varphi(s\sigma) - \varphi(t\sigma)) = 0. \end{aligned}$$

Therefore, $1 \otimes \partial_1^{F'}(\underline{t \approx s}) = 1 \otimes \partial_1^{F'}(\underline{t\sigma \approx s\sigma})$ in $\mathbb{Z}_d \otimes_{\mathcal{R}^{F'}} \text{im } \partial_1^{F'}$. Also, since $t\sigma \approx s\sigma \in E'$, we have $1 \otimes \partial_1^{F'}(\underline{t\sigma \approx s\sigma}) = \mathcal{H}(E' \rightarrow F')(1 \otimes \partial_1^{E'}(\underline{t\sigma \approx s\sigma}))$. Thus, $1 \otimes \partial_1^{F'}(\underline{t \approx s}) \in \text{im } \mathcal{H}(E' \rightarrow F')$. \blacktriangleleft

We show that Theorem 1 implies Theorem 5. Suppose R is a complete TRS with degree d . First, notice that if $d = 1$, then \mathbb{Z}_d is a trivial group and so is $\mathcal{H}(R)$. Hence Theorem 1 is not interesting in that case. We write $\check{\partial}_2^R$ for the map $\mathbb{Z}_d \otimes_{\mathcal{R}^R} \partial_2^R : \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_3^R \rightarrow \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R$ and write $\check{\partial}_1^R$ for the map $\mathbb{Z}_d \otimes (\partial_1^R : \mathcal{R}^R \underline{\mathbf{P}}_2^R \rightarrow \text{im } \partial_1^R)$. Since the sequence

$$\mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_3^R \xrightarrow{\check{\partial}_2^R} \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R \xrightarrow{\check{\partial}_1^R} \mathbb{Z}_d \otimes_{\mathcal{R}^R} \text{im } \partial_1^R \rightarrow 0$$

is exact, $\mathcal{H}(E) = \mathbb{Z}_d \otimes_{\mathcal{R}^R} \text{im } \partial_1^R$ is isomorphic to $\text{coker } \check{\partial}_2^R = \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R / \text{im } \check{\partial}_2^R$.

Let E be a set of equations with degree d' and R be a complete TRS with degree d such that $E^* \subset R^*$. We define $h : \mathbb{Z}_{d'} \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{\mathbf{P}}_2^E \rightarrow \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R$ by $h(1 \otimes \underline{t \approx s}) = 1 \otimes (\hat{t} - \hat{s})$.

► Lemma 27. $\check{\partial}_1^R \circ h = \mathcal{H}(E \rightarrow R) \circ \check{\partial}_1^E$. That is, the following diagram commutes:

$$\begin{array}{ccc} \mathbb{Z}_{d'} \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{\mathbf{P}}_2^E & \xrightarrow{\check{\partial}_1^E} & \mathbb{Z}_{d'} \otimes_{\mathcal{R}^E} \text{im } \partial_1^E \\ \downarrow h & & \downarrow \mathcal{H}(E \rightarrow R) \\ \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R & \xrightarrow{\check{\partial}_1^R} & \mathbb{Z}_d \otimes_{\mathcal{R}^R} \text{im } \partial_1^R. \end{array}$$

Proof. First, we show, by induction, $\check{\partial}_1^R(1 \otimes \hat{t}) = 1 \otimes (\varphi(\hat{t}) - \varphi(t)) \in \mathbb{Z}_d \otimes_{\mathcal{R}^R} \text{im } \partial_1^R$ for any term t . If $\hat{t} = 0$, or equivalently, t is normal, then the equality trivially holds. If $\hat{t} = C\underline{l \approx ru} + \hat{t}'$ ($C[l \circ u] = t$, $C[r \circ u] = t'$) and $\check{\partial}_1^R(1 \otimes \hat{t}') = 1 \otimes (\varphi(\hat{t}') - \varphi(t'))$, then $\check{\partial}_1^R(1 \otimes \hat{t}) = 1 \otimes (\varphi(\hat{t}) - \varphi(t) + \varphi(r) - \varphi(l))$. Since $1 \otimes (\varphi(r) - \varphi(l)) = 1 \otimes (C\varphi(r)u - C\varphi(l)u) = 1 \otimes (\varphi(t') - \varphi(t))$, we have $\check{\partial}_1^R(\hat{t}) = 1 \otimes (\varphi(\hat{t}) - \varphi(t))$.

Now, we have $\check{\partial}_1^R(h(1 \otimes \underline{t \approx s})) = \check{\partial}_1^R(1 \otimes \hat{t}) - \check{\partial}_1^R(1 \otimes \hat{s}) = 1 \otimes (\varphi(s) - \varphi(t))$ and thus $\mathcal{H}(E \rightarrow R)(\check{\partial}_1^E(1 \otimes \underline{t \approx s})) = 1 \otimes (\varphi(s) - \varphi(t))$. \blacktriangleleft

61:12 A Homological Condition on Equational Unifiability

The above lemma implies that the map

$$\bar{h} : \mathbb{Z}_{d'} \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{\mathbf{P}}_2^E / \ker \check{\partial}_1^E \rightarrow \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R / \ker \check{\partial}_1^R, \quad [x] \mapsto [h(x)]$$

is well-defined since if $x \in \ker \check{\partial}_1^E$, then $\check{\partial}_1^R(h(x)) = \mathcal{H}(E \rightarrow R)(\check{\partial}_1^E(x)) = 0$. Also, $\mathcal{H}(E \rightarrow R)$ is surjective iff \bar{h} is surjective since we have the diagram

$$\begin{array}{ccc} \mathbb{Z}_{d'} \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{\mathbf{P}}_2^E / \ker \check{\partial}_1^E & \xrightarrow{\cong} & \mathbb{Z}_{d'} \otimes_{\mathcal{R}^E} \text{im } \partial_1^E \\ \downarrow \bar{h} & & \downarrow \mathcal{H}(E \rightarrow R) \\ \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R / \ker \check{\partial}_1^R & \xrightarrow{\cong} & \mathbb{Z}_d \otimes_{\mathcal{R}^R} \text{im } \partial_1^R. \end{array}$$

Theorem 5 follows from Theorem 1 and the lemma below.

► **Lemma 28.** *The map $\mathcal{H}(E \rightarrow R)$ is surjective iff the matrix $(D(E)|U(E, R))$ is equivalent to $I_{n,m}$ and $n \leq m$ where n (resp. m) is the number of rows (resp. columns) in $(D(R)|U(E, R))$.*

Proof. We can see that $U(E, R)$ is a matrix representation of h and $D(R)$ is a matrix representation of $\check{\partial}_2^R$. So, $(D(E)|U(E, R))$ is equivalent to $I_{n,m}$ and $n \leq m$ iff the map

$$(\mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_3^R) \times (\mathbb{Z}_d \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{\mathbf{P}}_2^E) \rightarrow \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R, \quad (x, y) \mapsto \check{\partial}_2^R(x) + h(y)$$

is surjective.

Suppose $\mathcal{H}(E \rightarrow R)$ is surjective. Then, \bar{h} is surjective and so for any $z \in \mathbb{Z}_d \otimes_{\mathcal{R}^R} \mathcal{R}^R \underline{\mathbf{P}}_2^R$, we have $y \in \mathbb{Z}_{d'} \otimes_{\mathcal{R}^E} \mathcal{R}^E \underline{\mathbf{P}}_2^E$ and $z' \in \ker \check{\partial}_1^R$ satisfying $z = h(y) + z'$. Since $\ker \check{\partial}_1^R = \text{im } \check{\partial}_2^R$, there exists x such that $\check{\partial}_2^R(x) = z'$. Therefore, the map $(x, y) \mapsto \check{\partial}_2^R(x) + h(y)$ is surjective. The converse can be shown in a similar way. ◀

The above lemma implies that the necessary condition stated in Theorem 5 is independent of the choice of rewriting strategy. (\because The map $\mathcal{H}(E \rightarrow R)$ is defined independently from rewriting strategies.)

4.5 Functoriality

For a signature Σ , consider the category \mathcal{E}_Σ such that its objects are sets of equations over Σ and for each pair of objects E, E' with $E^* \subset E'^*$, there exists exactly one morphism $E \rightarrow E'$. Then, we shall see that $\mathcal{H} : \mathcal{E}_\Sigma \rightarrow \mathbf{Ab}$ is a functor. It is straightforward to show that $\mathcal{H}(E \rightarrow E)$ is an identity map, so we show

$$\mathcal{H}(E' \rightarrow E'') \circ \mathcal{H}(E \rightarrow E') = \mathcal{H}(E \rightarrow E'') \quad (4)$$

for any E, E', E'' with $E^* \subset E'^* \subset E''^*$. Recall that $\mathcal{H}(E \rightarrow E')$ is defined using the functor

$$\pi^{E, E'} : \mathcal{R}^E \rightarrow \mathcal{R}^{E'}, \quad [(C_1, u_1) + \cdots + (C_k, u_k)]^E \mapsto [(C_1, u_1) + \cdots + (C_k, u_k)]^{E'}.$$

For a set E'' of equations with $E'^* \subset E''^*$, we can see $\pi^{E', E''} \circ \pi^{E, E'} = \pi^{E, E''}$ and so

$$q^{d', d''} \otimes \bar{\pi}_{\mathbf{P}_1}^{E', E''} \circ q^{d, d'} \otimes \bar{\pi}_{\mathbf{P}_1}^{E, E'} = q^{d, d''} \otimes \bar{\pi}_{\mathbf{P}_1}^{E, E''}. \quad (5)$$

As we saw that the diagram (3) commutes, we have the commutative diagram

$$\begin{array}{ccc}
 \mathbb{Z}_d \otimes (\ker \partial_0^E(k)) & \xrightarrow{\zeta_k} & \mathbb{Z}_d \otimes_{\mathcal{R}^E} \ker \partial_0^E \\
 \downarrow q^{d,d'} \otimes \overline{\pi}_{\mathbf{P}_1}^{E,E'} & & \downarrow \mathcal{H}(E \rightarrow E') \\
 \mathbb{Z}_{d'} \otimes (\ker \partial_0^{E'}(k)) & \xrightarrow{\zeta_k} & \mathbb{Z}_{d'} \otimes_{\mathcal{R}^{E'}} \ker \partial_0^{E'} \\
 \downarrow q^{d',d''} \otimes \overline{\pi}_{\mathbf{P}_1}^{E',E''} & & \downarrow \mathcal{H}(E' \rightarrow E'') \\
 \mathbb{Z}_{d''} \otimes (\ker \partial_0^{E''}(k)) & \xrightarrow{\zeta_k} & \mathbb{Z}_{d''} \otimes_{\mathcal{R}^{E''}} \ker \partial_0^{E''}
 \end{array}
 \begin{array}{l}
 \\
 \mathcal{H}(E \rightarrow E'') \\
 \leftarrow
 \end{array}$$

where $d = \deg(E)$, $d' = \deg(E')$, and $d'' = \deg(E'')$. By (5) and by the uniqueness of $\mathcal{H}(E \rightarrow E'')$, we obtain the equality (4).

5 Related Work

5.1 Free Resolutions in Rewriting

The partial free resolution (2) was given by Malbos and Mimram in [11] to compute invariants called homology groups of an equational theory. For a signature Σ and set E of equational theory over Σ , if we have a free resolution $\dots \xrightarrow{\delta_3} F_3 \xrightarrow{\delta_2} F_2 \xrightarrow{\delta_1} F_1 \xrightarrow{\delta_0} F_0 \xrightarrow{\eta} \mathcal{Z}^E \rightarrow 0$ of \mathcal{Z}^E , the i -th homology group $H_i(\Sigma, E)$ is defined as the abelian group $\ker(\mathbb{Z}_d \otimes \delta_{i-1}) / \text{im}(\mathbb{Z}_d \otimes \delta_i)$. As a general fact of homological algebra, it is shown that the homology groups do not depend on the choice of free resolution. Also, if E' is a set of equations over Σ' and (Σ', E') is Tietze equivalent (see [11] for the definition) to (Σ, E) , $H(\Sigma', E')$ is isomorphic to $H(\Sigma, E)$. The partial free resolution (2) is useful to compute the homology groups since each generating set \mathbf{P}_i is finite. Also, it is shown that for any signature Σ' and set E' of equations over Σ' , if (Σ', E') is Tietze equivalent to (Σ, E) , E' has at least $s(H_2(\Sigma, E))$ elements where $s(A)$ is the minimum number of generators of A . In [8], the author showed that for a set E' of equations over Σ which E is also over, if E' is equivalent to E (in the sense $E^* = E'^*$), E' has at least $s(H_2(\Sigma, E)) + s(\text{im}(\mathbb{Z}_d \otimes \partial_1))$ elements.

Homology groups are defined for many mathematical objects. Homology groups of a group, also called group homologies, have a close relationship with homology groups of an equational theory. For a group G , its homology $H_i(G)$ is defined as follows. Consider the group ring $\mathbb{Z}\langle G \rangle$ and a free resolution of \mathbb{Z} as a left $\mathbb{Z}\langle G \rangle$ -module

$$\dots \xrightarrow{\delta_3} F_3 \xrightarrow{\delta_2} F_2 \xrightarrow{\delta_1} F_1 \xrightarrow{\delta_0} F_0 \xrightarrow{\eta} \mathbb{Z} \rightarrow 0,$$

then $H_i(G) = \ker(\mathbb{Z} \otimes \delta_{i-1}) / \text{im}(\mathbb{Z} \otimes \delta_i)$. If a group G is presented by some generators $S = \{g_1, g_2, \dots\}$ and relations $T = \{r_1 = 1, r_2 = 1, \dots\}$, it is known that there is a partial free resolution

$$\mathbb{Z}\langle G \rangle \underline{T} \rightarrow \mathbb{Z}\langle G \rangle \underline{S} \rightarrow \mathbb{Z}\langle G \rangle \rightarrow \mathbb{Z} \rightarrow 0.$$

(See [3, Exercise 3 in §II.5] for example.)

In [14], Squier considered free resolutions of \mathbb{Z} as a module over the monoid ring $\mathbb{Z}\langle M \rangle$ for a monoid M . Also in this case, if M is presented by generators $S = \{g_1, g_1, \dots\}$ and relations $T = \{l_1 = r_1, l_2 = r_2, \dots\}$, we have a partial free resolution

$$\mathbb{Z}\langle M \rangle \underline{T} \rightarrow \mathbb{Z}\langle M \rangle \underline{S} \rightarrow \mathbb{Z}\langle M \rangle \rightarrow \mathbb{Z} \rightarrow 0.$$

Moreover, he showed that if the relations form a complete string rewriting system, the partial free resolution is extended to

$$\mathbb{Z}\langle M \rangle \underline{U} \rightarrow \mathbb{Z}\langle M \rangle \underline{T} \rightarrow \mathbb{Z}\langle M \rangle \underline{S} \rightarrow \mathbb{Z}\langle M \rangle \rightarrow \mathbb{Z} \rightarrow 0.$$

where U is the set of critical pairs. This resolution inspired our free resolution (2) for an equational theory.

5.2 Narrowing

For a TRS R , a term s is said to be *narrowable* into a term t if there exist a rule $l \rightarrow r \in R$, a context C , and non-variable term s' such that $s = C[s']$, s' and l are unifiable with the mgu σ , and $t = C[r]\sigma$. (We rename variables in l so that $\text{Var}(l) \cup \text{Var}(s) = \emptyset$.) In that case, we write $s \rightsquigarrow_{\sigma, R} t$. The sequence $t_0 \rightsquigarrow_{\sigma_1, R} t_1 \rightsquigarrow_{\sigma_2, R} \cdots \rightsquigarrow_{\sigma_n, R} t_n$ is abbreviated to $t_0 \rightsquigarrow_{\sigma, R}^* t_n$ for $\sigma = \sigma_0 \sigma_1 \dots \sigma_n$. For two substitutions σ, θ and a set X of variables, σ is *more general modulo R on X* than θ , denoted $\sigma \leq_R^X \theta$, if there exists a substitution τ such that $x\theta \approx_R x\sigma\tau$ for any $x \in X$. Then, it is known that narrowing is a complete procedure for R -unification:

- **Theorem 29** ([7]). *Suppose that R is complete and eq be a new symbol with arity 2.*
- *If $\text{eq}(s, t) \rightsquigarrow_{\sigma, R}^* \text{eq}(s', t')$ and s', t' are unifiable with the mgu τ , s, t are R -unifiable with the unifier $\sigma\tau$.*
 - *If s, t are R -unifiable with a unifier θ , then there exist a narrowing sequence $\text{eq}(s, t) \rightsquigarrow_{\sigma, R}^* \text{eq}(s', t')$ and an mgu τ of s', t' such that $\sigma\tau \leq_R^{\text{Var}(\text{eq}(s, t))} \theta$.*

Consider Example 1 again. We can say $x_1 + a$ and $x_1 + b$ are not E_1 -unifiable since $\text{eq}(x_1 + a, x_1 + b)$ is not narrowable by any rules in E_1 .

For Example 2, however, we have an infinite narrowing sequence from $\text{eq}(x_1 + x_1, s(0))$:

$$\begin{aligned} \text{eq}(x_1 + x_1, s(0)) &\rightsquigarrow_{x_1 \mapsto s(x_1), E_2} \text{eq}(s(x_1 + s(x_1)), s(0)) \\ &\rightsquigarrow_{x_1 \mapsto s(x_1), E_2} \text{eq}(s(s(x_1 + s(x_1))), s(0)) \\ &\rightsquigarrow_{x_1 \mapsto s(x_1), E_2} \cdots \end{aligned}$$

so we can see that narrowing is a semi-decision procedure of the problem of equational unification. It has been studied that what kind of restriction on a TRS ensures termination of narrowing [1].

6 Conclusion

We have obtained a functor $\mathcal{H} : \mathcal{E}_\Sigma \rightarrow \mathbf{Ab}$ where \mathcal{E}_Σ is the category of sets of equations and proved that E -unifiability of two terms t, s implies the surjectivity of the homomorphism $\mathcal{H}(E \rightarrow E \cup \{t \approx s\})$. In case where $E \cup \{t \approx s\}$ has a complete TRS, the surjectivity of $\mathcal{H}(E \rightarrow E \cup \{t \approx s\})$ is equivalent to the condition that the matrix $(D(R)|U(E, R))$ has full rank. Therefore, our theorem gives a sound procedure for checking non- E -unifiability.

References

- 1 María Alpuente, Santiago Escobar, and José Iborra. Termination of narrowing revisited. *Theoretical Computer Science*, 410(46):4608–4625, 2009. Abstract Interpretation and Logic Programming: In honor of professor Giorgio Levi.
- 2 Franz Baader, Wayne Snyder, Paliath Narendran, Manfred Schmidt-Schauss, and Klaus Schulz. Unification theory. In *Handbook of Automated Reasoning*, Handbook of Automated Reasoning, pages 445–533. North-Holland, Amsterdam, 2001.

- 3 K. S. Brown. *Cohomology of Groups*, volume 87 of *Graduate Texts in Mathematics*. Springer-Verlag New York, 1982.
- 4 W. Brown. *Matrices over commutative rings*. M. Dekker, New York, 1993.
- 5 Martin Davis. Hilbert’s tenth problem is unsolvable. *The American Mathematical Monthly*, 80(3):233–269, 1973. doi:10.1080/00029890.1973.11993265.
- 6 M. Fay. First-order unification in an equational theory. In *4th Workshop on Automated Deduction*, Austin, Texas, 1978.
- 7 Jean-Marie Hullot. Canonical forms and unification. In Wolfgang Bibel and Robert Kowalski, editors, *5th Conference on Automated Deduction Les Arcs, France, July 8–11, 1980*, pages 318–334, Berlin, Heidelberg, 1980. Springer Berlin Heidelberg.
- 8 Mirai Ikebuchi. A Lower Bound of the Number of Rewrite Rules Obtained by Homological Methods. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- 9 M. Jantzen. A note on a special one-rule semi-thue system. *Information Processing Letters*, 21(3):135–140, 1985.
- 10 Mamuka Jibladze and Teimuraz Pirashvili. Cohomology of algebraic theories. *Journal of Algebra*, 137(2):253–296, 1991.
- 11 P. Malbos and S. Mimram. Homological computations for term rewriting systems. In *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*, volume 52 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- 12 Ju V Matijasevic. Enumerable sets are diophantine. In *Soviet Math. Dokl.*, volume 11, pages 354–358, 1970.
- 13 B. Mitchell. Rings with several objects. *Advances in Mathematics*, 8(1):1–161, 1972.
- 14 C. C. Squier. Word problems and a homological finiteness condition for monoids. *Journal of Pure and Applied Algebra*, 49(1-2):201–217, 1987.
- 15 Charles A. Weibel. *An Introduction to Homological Algebra*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1994.

A The matrix for $E_1 \cup \{x_1 + x_1 \approx 0\}$

The TRS $E_1 \cup \{x_1 + x_1 \approx 0\}$ has the following complete TRS R_3 :

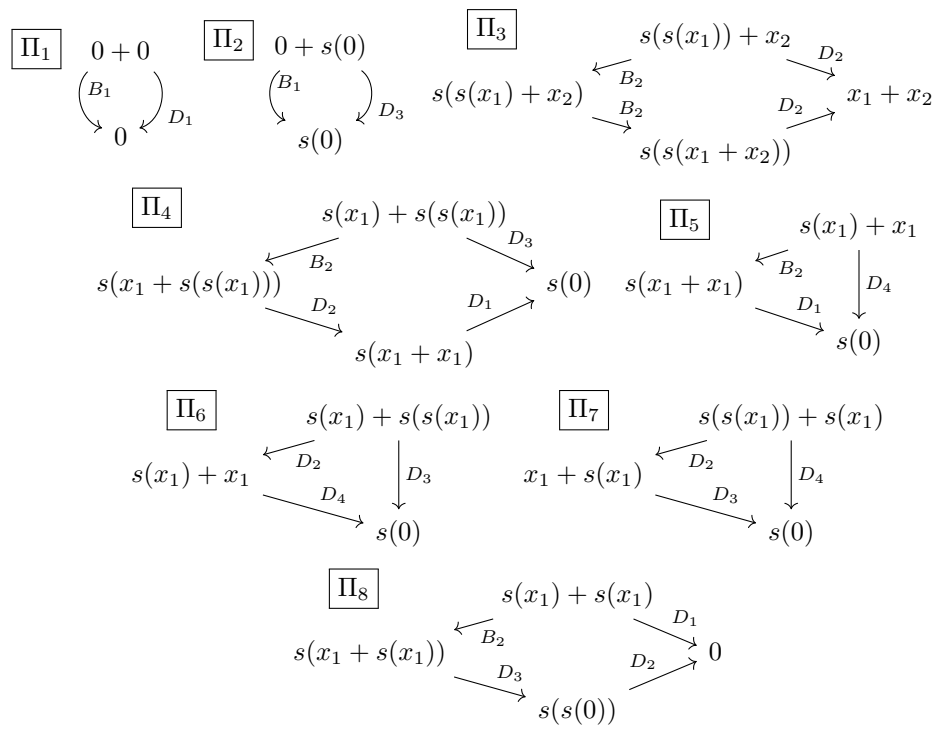
$$\begin{array}{llll} B_1 : 0 + x_1 \rightarrow x_1 & B_2 : s(x_1) + x_2 \rightarrow s(x_1 + x_2) & D_1 : x_1 + x_1 \rightarrow 0 \\ D_2 : s(s(x_1)) \rightarrow x_1 & D_3 : x_1 + s(x_1) \rightarrow s(0) & D_4 : s(x_1) + x_1 \rightarrow s(0). \end{array}$$

The critical pairs are listed in Fig. 1 and the matrix $(D(R_3)|U(E_1, R_3))$ is given as follows.

$$\begin{array}{cccccccccc|cc} & \Pi_1 & \Pi_2 & \Pi_3 & \Pi_4 & \Pi_5 & \Pi_6 & \Pi_7 & \Pi_8 & \Pi_9 & B_1 & B_2 \\ \begin{array}{l} B_1 \\ B_2 \\ D_1 \\ D_2 \\ D_3 \\ D_4 \end{array} & \left(\begin{array}{cccccccccc|cc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

It is not too hard to check that it has full rank.

61:16 A Homological Condition on Equational Unifiability



■ **Figure 1** Critical pairs of R_3 in Example 3.

Ordered Fragments of First-Order Logic

Reijo Jaakkola  

University of Helsinki, Finland
Tampere University, Finland

Abstract

Using a recently introduced algebraic framework for classifying fragments of first-order logic, we study the complexity of the satisfiability problem for several ordered fragments of first-order logic, which are obtained from the ordered logic and the fluted logic by modifying some of their syntactical restrictions.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases ordered logic, fluted logic, complexity, decidability

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.62

Related Version *Full Version*: <https://arxiv.org/abs/2103.08046>

Funding The research leading to this work was supported by the Academy of Finland grants 324435 and 328987.

Acknowledgements The author wishes to thank Antti Kuusisto for many helpful discussions on fragments of first-order logic. The author also wishes to thank the anonymous reviewers for their valuable suggestions concerning the exposition of the motivation and the context of the present work.

1 Introduction

The study of computational properties of fragments of first-order logic is an active research area, which has been motivated by the general observation that most of the logics used in computer science applications, such as the description logics, can be translated into first-order logic [5]. The main goal of this area is to discover expressive fragments which have nice computational properties; in particular, their satisfiability problem – the problem of determining whether a given sentence of the fragment is satisfiable – should be decidable. Perhaps the most widely studied decidable fragments of first-order logic are the two-variable logic FO^2 and the guarded fragment GF, and their various extensions, see for example [2, 3, 11, 17]. Recently there has been an increasing interest on studying fragments that we refer to in this paper collectively as the *ordered fragments* [1, 12, 13, 14].

Informally speaking, we define a fragment of first-order logic to be ordered, if the syntax of the fragment restricts permutations of variables (with respect to some ordering of the variables) and the order in which the variables are to be quantified. To illustrate these restrictions, consider the sentence

$$\forall v_1(P(v_1) \rightarrow \exists v_2(R(v_1, v_2) \wedge \forall v_3 S(v_1, v_2, v_3))).$$

This sentence is ordered in the sense that variables occur in the right order in the atomic formulas, and they are quantified in the correct order. This particular sentence belongs to the most well-known member of this family of logics, namely the so-called fluted logic, which was proved to have a TOWER-complete satisfiability problem in [13].

Another important ordered fragment, which is also relevant for the present work, is the so-called ordered logic, which on the level of sentences is a fragment of fluted logic (for a formal definition of this logic we refer the reader to section 3). In [4] it was proved that the



© Reijo Jaakkola;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 62; pp. 62:1–62:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

complexity of the satisfiability problem of this logic is in PSPACE, by reducing this problem to the satisfiability problem of modal logic over serial frames. It turns out that the satisfiability problem of this fragment is also PSPACE-complete, and the proof for PSPACE-hardness can be found in the full version of this paper.

Thus the aforementioned syntactical restrictions, which guarantee that the formulas of the fragment are ordered, seem to guarantee that the underlying fragments are decidable. Another aspect that makes the ordered fragments of first-order logic interesting is that they are orthogonal in expressive power with respect to other well-known fragments of first-order logic, such as the guarded fragments. For instance, the formula $\forall v_1 \forall v_2 \forall v_3 R(v_1, v_2, v_3)$ is clearly ordered, but it expresses a property that is, for example, neither expressible in GF nor in FO^2 . Thus they form a genuinely new family of decidable fragments of first-order logic, and hence they provide us with a fresh perspective on the question of what makes a fragment of first-order logic decidable.

Ordered fragments can also be used to tame the complexity of decidable fragments. To give an example of what we mean by this, we mention the recent work conducted in [1] where the author showed, among other results, that even though the complexity of the satisfiability problem of GF is 2EXPTIME -complete, it becomes EXPTIME -complete if we restrict attention to the set of formulas that also belong to the fluted logic. More precisely the author introduced a new ordered fragment, namely the forward guarded fragment which contains as a proper subset the aforementioned intersection of GF and the fluted logic, and then proceeded to prove that the satisfiability problem of this stronger logic is EXPTIME -complete.

Since the syntax of ordered logics restricts heavily the permutations of variables and the order in which the variables are quantified, their syntax can often be presented naturally in a variable-free way. Indeed, the fluted logic was originally discovered by Quine as a by-product of his attempts to present the full syntax of first-order logic in a variable-free way by using the predicate functor logic [15, 16]. Interestingly, this approach was also adopted in the recent papers [12, 14], where the fluted logic was presented using its variable-free syntax.

Recently a research program was introduced in [6, 7, 10] for classifying fragments of first-order logic within an algebraic framework that is closely related to the aforementioned predicate functor logic. In a nutshell, the basic idea is to identify fragments of first-order logic with finite algebraic signatures (for more details, see the next section). The algebraic framework naturally suggests the idea of defining logics with limited permutations, and hence it is well suited for defining various logics that belong to the family of ordered fragments.

The main purpose of the present work is to apply the aforementioned algebraic framework to study how the complexities of ordered and fluted logic change, if we modify their syntax in various ways. The first question that we study in this paper is whether one could extend the syntax of ordered logic while maintaining the requirement that the complexity of the satisfiability problem remains relatively low. We will formalize different minimal extension of the ordered logic using additional algebraic operators and study the complexities of the resulting logics. The picture that emerges from our results seems to suggest that even if one modifies the syntax of the ordered logic in a very minimal way, the resulting logics will most likely have much higher complexity. For instance, if we relax even slightly the order in which the variables can be quantified, the resulting logic will have NEXPTIME -hard satisfiability problem. However, there are also exceptions to this rule, since the complexity of ordered logic with equality turns out to be the same as the complexity of the regular ordered logic.

Motivated by the recent study of one-dimensional guarded fragments conducted in [8], we will also study the one-dimensional fragments of fluted logic and ordered logic. Intuitively a logic is called one-dimensional if quantification is limited to applications of blocks of existential

(universal) quantifiers such that at most one variable remains free in the quantified formula. Imposing the restriction of one-dimensionality to fluted logic and ordered logic decreases quite considerably the complexity of the underlying logics: the complexity of the one-dimensional fluted logic is NEXPTIME-complete while the complexity of the one-dimensional ordered logic (even with equality) is NP-complete. In the case of one-dimensional fluted logic we are able to add some further algebraic operators into its syntax without increasing its complexity.

We will also prove that several natural extensions of the ordered logic and the fluted logic are undecidable. First, for the ordered logic we are able to show that if we allow variables to be quantified in an arbitrary order, then the resulting logic is undecidable. Secondly, we are able to show that if we lift the restrictions on how the variables in the atomic formulas can be permuted in the one-dimensional fluted logic, then the resulting logic is undecidable. Finally, in the case of the full fluted logic, we can show that if we relax only slightly the way variables can be permuted and the order in which variables can be quantified, then the resulting logic is undecidable.

2 Algebraic way of presenting logics

The purpose of this section is to present the algebraic framework introduced in [6, 7, 10] for defining logics in an algebraic way. We will be working with purely relational vocabularies with no constants and function symbols. In addition we will not consider vocabularies with 0-ary relation symbols. Throughout this paper we will use the convention where the domain of a model \mathfrak{A} will be denoted by the set A .

Let A be an arbitrary set. As usual, a k -**tuple** over A is an element of A^k . We will use ϵ to denote the 0-ary tuple. Given a non-negative integer k , a k -ary **AD-relation** over A is a pair $T = (X, k)$, where $X \subseteq A^k$. Here 'AD' stands for arity-definite. Given a k -ary AD-relation $T = (X, k)$ over A , we will use $(a_1, \dots, a_k) \in T$ to denote $(a_1, \dots, a_k) \in X$. Given an AD-relation T , we will use $ar(T)$ to denote its arity.

Given a set A , we will use $AD(A)$ to denote the set of all AD-relations over A . If $T_1, \dots, T_k \in AD(A)$, then the tuple (A, T_1, \dots, T_k) will be called an **AD-structure** over A . A bijection $g : A \rightarrow B$ is an **isomorphism** between AD-structures (A, T_1, \dots, T_k) and (B, S_1, \dots, S_k) , if for every $1 \leq \ell \leq k$ we have that $ar(T_\ell) = ar(S_\ell)$, and g is an ordinary isomorphism between the relational structures $(A, rel(T_1), \dots, rel(T_k))$ and $(B, rel(S_1), \dots, rel(S_k))$, where $rel(T)$ denotes the underlying relation of an AD-relation.

The following definition was introduced in [7], where it was called arity-regular relation operator.

► **Definition 1.** A k -ary **relation operator** F is a mapping which associates to each set A a function $F^A : AD(A)^k \rightarrow AD(A)$ and which satisfies the following requirements.

1. The operator F is isomorphism invariant in the sense that whenever two AD-structures (A, T_1, \dots, T_k) and (B, S_1, \dots, S_k) are isomorphic via g , the same mapping is also an isomorphism between the AD-structures $(A, F^A(T_1, \dots, T_k))$ and $(B, F^B(S_1, \dots, S_k))$.
2. There exists a function $\sharp : \mathbb{N}^k \rightarrow \mathbb{N}$ so that for every AD-structure (A, T_1, \dots, T_k) we have that the arity of the AD-relation $F^A(T_1, \dots, T_k)$ is $\sharp(ar(T_1), \dots, ar(T_k))$. In other words the arity of the output AD-relation is always determined fully by the sequence of arities of the input AD-relations.

Given a set of relation operators \mathcal{F} and a vocabulary τ , we can define a language $GRA(\mathcal{F})[\tau]$ as follows, where $R \in \tau$ and $F \in \mathcal{F}$:

$$\mathcal{T} ::= \perp \mid \top \mid R \mid F(\underbrace{\mathcal{T}, \dots, \mathcal{T}}_{ar(F) \text{ times}}).$$

Here “GRA” stands for general relational algebra. We sometimes use the infix notation instead of the prefix notation, if the infix notation is more conventional. Furthermore we will drop the brackets in the case where F is unary operator.

If the underlying vocabulary τ is clear from context or irrelevant, we will write $\text{GRA}(\mathcal{F})$ instead of $\text{GRA}(\mathcal{F})[\tau]$. The members of $\text{GRA}(\mathcal{F})$ will be referred to as **terms**. In the case where \mathcal{F} is a finite set $\{F_1, \dots, F_n\}$, we will use $\text{GRA}(F_1, \dots, F_n)$ to denote $\text{GRA}(\{F_1, \dots, F_n\})$.

Given a model \mathfrak{A} of vocabulary τ and $\mathcal{T} \in \text{GRA}(\mathcal{F})[\tau]$, we define its **interpretation** $\llbracket \mathcal{T} \rrbracket_{\mathfrak{A}}$ recursively as follows:

1. If $\mathcal{T} = \perp$, then we define $\llbracket \mathcal{T} \rrbracket_{\mathfrak{A}} := (\emptyset, 0)$, and if $\mathcal{T} = \top$, then we define $\llbracket \mathcal{T} \rrbracket_{\mathfrak{A}} := (\{\epsilon\}, 0)$.
2. If $\mathcal{T} = R \in \tau$, then we define $\llbracket R \rrbracket_{\mathfrak{A}} = (R^{\mathfrak{A}}, ar(R))$.
3. If $\mathcal{T} = F(\mathcal{T}_1, \dots, \mathcal{T}_k)$, then we define $\llbracket \mathcal{T} \rrbracket_{\mathfrak{A}} = F_A(\llbracket \mathcal{T}_1 \rrbracket_{\mathfrak{A}}, \dots, \llbracket \mathcal{T}_k \rrbracket_{\mathfrak{A}})$.

Note that the interpretation of a term over \mathfrak{A} is an AD-relation over A . The arity of this AD-relation (over some fixed model) is called the arity of the term \mathcal{T} and we will denote it by $ar(\mathcal{T})$. Note that by definition the arity of the output relation is independent of the underlying model, which guarantees that $ar(\mathcal{T})$ is well-defined.

Given two k -ary terms \mathcal{T} and \mathcal{P} over the same vocabulary, we say that \mathcal{T} is **contained** in \mathcal{P} , if for every model \mathfrak{A} over τ and for every $(a_1, \dots, a_k) \in A^k$ we have that if $(a_1, \dots, a_k) \in \llbracket \mathcal{T} \rrbracket_{\mathfrak{A}}$ then $(a_1, \dots, a_k) \in \llbracket \mathcal{P} \rrbracket_{\mathfrak{A}}$. We will denote this by $\mathcal{T} \models \mathcal{P}$. If \mathcal{T} is a 0-ary term and \mathfrak{A} is a model so that $\llbracket \mathcal{T} \rrbracket_{\mathfrak{A}} = (\{\epsilon\}, 0)$, then we denote this by $\mathfrak{A} \models \mathcal{T}$. Given a 0-ary term \mathcal{T} , we say that \mathcal{T} is **satisfiable** if there exists a model \mathfrak{A} so that $\mathfrak{A} \models \mathcal{T}$.

We will conclude this section by briefly indicating how we can compare the expressive power of algebras with fragments of FO. Let $k \geq 0$ and consider an FO-formula $\varphi(v_{i_1}, \dots, v_{i_k})$, where $(v_{i_1}, \dots, v_{i_k})$ lists all the free variables of φ , and $i_1 < \dots < i_k$. If \mathfrak{A} is a suitable model, then φ defines the AD-relation $\llbracket \varphi \rrbracket_{\mathfrak{A}} := (\{(a_1, \dots, a_k) \mid \mathfrak{A} \models \varphi(a_1, \dots, a_k)\}, k)$ over A . Given a k -ary term \mathcal{T} and FO-formula $\varphi(v_{i_1}, \dots, v_{i_k})$ over the same vocabulary, we say that \mathcal{T} is **equivalent** with φ if for every model \mathfrak{A} we have that $\llbracket \mathcal{T} \rrbracket_{\mathfrak{A}} = \llbracket \varphi \rrbracket_{\mathfrak{A}}$.

If \mathcal{F} is a set of relation operators and $\mathcal{L} \subseteq \text{FO}$, then we say that $\text{GRA}(\mathcal{F})$ and \mathcal{L} are **equivalent**, if for every $\mathcal{T} \in \text{GRA}(\mathcal{F})$ there exists an equivalent formula $\varphi \in \mathcal{L}$, and conversely for every formula $\varphi \in \mathcal{L}$ there exists an equivalent term $\mathcal{T} \in \text{GRA}(\mathcal{F})$. Similarly, we say that $\text{GRA}(\mathcal{F})$ and \mathcal{L} are **sententially equivalent**, if for every 0-ary term $\mathcal{T} \in \text{GRA}(\mathcal{F})$ there exists an equivalent sentence $\varphi \in \mathcal{L}$, and conversely for every sentence $\varphi \in \mathcal{L}$ there exists an equivalent 0-ary term $\mathcal{T} \in \text{GRA}(\mathcal{F})$.

3 Relevant fragments and complexity results

The purpose of this section is to present the relevant FO-fragments that we are going to study and to present the main complexity results that we are able to obtain. Through out this section (X, k) and (Y, ℓ) are AD-relations over some set A .

We are going to start by defining formally the ordered logic OL, which will form the backbone for the rest of fragments studied in this paper.

► **Definition 2.** Let $\bar{v}_\omega = (v_1, v_2, \dots)$ and let τ be a vocabulary. For every $k \in \mathbb{N}$ we define the set $\text{OL}^k[\tau]$ as follows.

1. Let $R \in \tau$ be an ℓ -ary relation symbol and consider the prefix (v_1, \dots, v_ℓ) of \bar{v}_ω containing precisely ℓ -variables. If $k \geq \ell$, then $R(v_1, \dots, v_\ell) \in \text{OL}^k[\tau]$.
2. Let $\ell \leq \ell' \leq k$ and suppose that $\varphi \in \text{OL}^\ell[\tau]$ and $\psi \in \text{OL}^{\ell'}[\tau]$. Then $\neg\varphi, (\varphi \wedge \psi) \in \text{OL}^k[\tau]$.
3. If $\varphi \in \text{OL}^{k+1}[\tau]$, then $\exists v_{k+1} \varphi \in \text{OL}^k[\tau]$.

Finally we define $\text{OL}[\tau] := \bigcup_k \text{OL}^k[\tau]$.

► **Remark 3.** The way we have presented the syntax of OL here is slightly different from the way it is often presented in the literature. The two logics are nevertheless equivalent on the level of sentences.

The syntax of this logic is somewhat involved, but it can be given a very nice algebraic characterization using just three relation operators $\{\neg, \cap, \exists\}$, which we are going to define next. Recalling that if F is a relation operator, then F^A denotes the function to which F maps the set A , we can define the relation operators as follows.

- ¬) We define $\neg^A(X, k) = (A^k \setminus X, k)$. We call \neg the **complementation** operator.
- ∩) If $k \neq \ell$, then we define $\cap^A((X, k), (Y, \ell)) = (\emptyset, 0)$. Otherwise we define

$$\cap^A((X, k), (Y, \ell)) = (X \cap Y, k).$$

We call \cap the **intersection** operator.

- ∃) If $k = 0$, then we define $\exists^A(X, k) = (X, k)$. Otherwise we define

$$\exists^A(X, k) = (\{(a_1, \dots, a_k) \mid (a_1, \dots, a_k, b) \in X, \text{ for some } b \in A\}, k - 1).$$

We call \exists the **projection** operator.

The following proposition establishes the promised characterization result.

► **Proposition 4.** *OL and $\text{GRA}(\neg, \cap, \exists)$ are sentimentally equiexpressive.*

The complexity of OL is rather low and thus it is natural to ask how it changes if we add additional operators to the syntax of the logic. The first operator that is studied in this paper is the operator E , which we define as follows.

- E) If $k < 2$, then we define $E^A(X, k) = (X, k)$. Otherwise we define

$$E^A(X, k) = (\{(a_1, \dots, a_k) \in X \mid a_{k-1} = a_k\}, k).$$

We call E the **equality** operator.

It turns out that the addition of equality does not increase the complexity of ordered logic. In our proof for the PSPACE upper bound, it will be convenient to extend the ordered logic with an additional operator I , which we define as follows.

- I) If $k \leq 1$, then we define $I^A(X, k) = (X, k)$, and otherwise we define

$$I^A(X, k) = (\{(a_1, \dots, a_{k-1}) \in A^{k-1} \mid (a_1, \dots, a_{k-1}, a_{k-1}) \in X\}, k - 1).$$

We call I the substitution operator.

In contrast with the equality operator, adding either of the following two operators to OL will result in a logic with NEXPTIME-hard satisfiability problem.

- s) If $k < 2$, then we define $s^A(X, k) = (X, k)$. Otherwise we define

$$s^A(X, k) = (\{(a_1, \dots, a_{k-2}, a_k, a_{k-1}) \mid (a_1, \dots, a_k) \in X\}, k).$$

We call s the **swap** operator.

C) If $k \neq 1$ and $\ell \leq 1$, then we define $C^A((X, k), (Y, \ell)) = (\emptyset, 0)$. In the case where $1 = k \leq \ell$ (the case $1 = \ell \leq k$ is defined similarly) we will define

$$C^A((Y, \ell), (X, k)) = (\{(a_1, \dots, a_\ell) \in Y \mid a_\ell \in X\}, \ell).$$

We call C the **one-dimensional intersection**.

The intuition behind the swap operator is clear: it lifts in a minimal way the ordering restriction on the syntax of ordered logic. The one-dimensional intersection may appear to be somewhat unnatural, but the underlying intuition is that we want to lift the uniformity imposed by \cap in a minimal way.

The other ordered fragment investigated in this paper is the fluted logic FL. We will not give a formal definition for this fragment here, but instead we will introduce its algebraic characterization using the operators $\{\neg, \dot{\cap}, \exists\}$, where $\dot{\cap}$ is defined as follows.

$\dot{\cap}$) If $m := \max\{k, \ell\}$, then we define

$$\dot{\cap}^A((X, k), (Y, \ell)) = (\{(a_1, \dots, a_m) \mid (a_{m-k+1}, \dots, a_m) \in X \text{ and } (a_{m-\ell+1}, \dots, a_m) \in Y\}, m),$$

We call $\dot{\cap}$ the **suffix intersection**.

The following result was proved in [7].

► **Proposition 5.** FL and $\text{GRA}(\neg, \dot{\cap}, \exists)$ are equiexpressive.

It was proved in [13] that the satisfiability problem for FL is TOWER-complete. The natural follow-up question is then to study what fragments of FL have more feasible complexity. In this paper we approach this question by studying the so-called *one-dimensional* fragment of fluted logic. To give this logic an algebraic characterization, we will need to introduce two additional operators, \exists_1 and \exists_0 , which we define as follows.

\exists_1) If $k < 2$, then we define $\exists_1^A(X, k) = (X, k)$. Otherwise we define

$$\exists_1^A(X, k) = (\{a \in A \mid \text{There exists } \bar{b} \in A^{k-1} \text{ such that } a\bar{b} \in X\}, 1)$$

\exists_0) If $k = 0$, then we define $\exists_0^A(X, k) = (X, k)$. Otherwise we define $\exists_0^A(X, k)$ to be $(\{\epsilon\}, 0)$, if X is non-empty, and $(\emptyset, 0)$, if X is empty.

We call collectively the operators \exists_1 and \exists_0 **one-dimensional projection** operators. These operators correspond to quantification which leaves at most one free-variable free. Now we define the algebra $\text{GRA}(\neg, \dot{\cap}, \exists_1, \exists_0)$ to be the one-dimensional fluted logic.

As one might expect, imposing the one-dimensionality requirement to formulas of FL will result in a logic with much lower complexity. The exact complexity of one-dimensional FL turns out to be NEXPTIME-complete, even for its extension with the swap and equality operators $\text{GRA}(s, E, \neg, \dot{\cap}, \exists_1, \exists_0)$. In this paper we also study the one-dimensional fragment of ordered logic with equality operator $\text{GRA}(E, \neg, \cap, \exists)$, for which the satisfiability problem turns out to be just NP-complete.

Besides just decidability results, we will also prove several undecidability results. To state some of these results, we will first define the following operator p .

p) If $k < 2$, then we define $p^A(X, k) = (X, k)$. Otherwise we define

$$p^A(X, k) = (\{(a_1, \dots, a_k) \mid (a_k, a_1, \dots, a_{k-1}) \in X\}, k).$$

We call p the **cyclic permutation** operator.

■ **Table 1** Complexities of the fragments. For each of the fragments the satisfiability problem is complete for the corresponding complexity class listed in the second column, excluding the case $\{s, E, \neg, C, \cap, \exists\}$ where the complexity is not known. All the results listed here are new.

$E, \neg, \cap, \exists_1, \exists_0$	NP
E, \neg, \cap, \exists	PSPACE
$s, \neg, C, \cap, \exists$	NEXPTIME
$E, \neg, C, \cap, \exists$	NEXPTIME
$s, E, \neg, C, \cap, \exists$?
$s, E, \neg, \hat{\cap}, \exists_1, \exists_0$	NEXPTIME
p, \neg, \cap, \exists	Π_1^0
$p, \neg, \hat{\cap}, \exists_1, \exists_0$	Π_1^0
$s, \neg, \hat{\cap}, \exists$	Π_1^0

Adding p to an ordered fragments correspondence essentially to the removal of the syntactical restriction that variables should be quantified in a specific order. The following theorem collects our undecidability results.

► **Theorem 6.** *Suppose that \mathcal{F} is a set of relation operators that contains $\{p, \neg, \cap, \exists\}$, $\{p, \neg, \hat{\cap}, \exists_1, \exists_0\}$ or $\{s, \neg, \hat{\cap}, \exists\}$. Now the satisfiability problem for $\text{GRA}(\mathcal{F})$ is Π_1^0 -hard.*

Let us conclude this section by mentioning briefly two complexity results that follow immediately from the literature and which complement the picture emerging from the results listed in Table 1. First, it is easy to verify that $\text{GRA}(p, s, E, \neg, C, \cap, \exists_1, \exists_0)$ is essentially equivalent with one-dimensional uniform fragment UF_1 , which was proved to be NEXPTIME-complete in [9]. The second result that we should mention is that the satisfiability problem for $\text{GRA}(E, \neg, \hat{\cap}, \exists)$ is TOWER-complete, since it contains FL and it can be translated to FL with equality, for which the satisfiability problem was recently proved in [14] to be TOWER-complete.

4 Tables and normal forms

In this paper we are going to perform several model constructions and hence it is useful to start by collecting some definitions and tools that we are going to need in the later sections.

► **Definition 7.** *Let $k \in \mathbb{Z}_+$ and $\mathcal{F} \subseteq \{I, s\}$. A k -table with respect to \mathcal{F} is a maximally consistent set of k -ary terms of the form \mathcal{T} or $\neg\mathcal{T}$, where $\mathcal{T} \in \text{GRA}(\mathcal{F})$. Given a model \mathfrak{A} and $\bar{a} \in A^k$, we will use $\text{tp}_{\mathfrak{A}}(\bar{a})$ to denote the k -table realized by \bar{a} .*

We will identify k -tables ρ with the terms $\bigcap_{\alpha \in \rho} \alpha$, which makes sense since all of the algebraic signatures that we are going to consider always include the operator \cap . This allows us to use notation such as $\rho \models \rho'$, where ρ and ρ' are k -tables. Furthermore, we will refer to 1-tables also as 1-types. We say that $a \in A$ is **king**, if there is no other element in the model that realizes the same 1-type.

Notice that there is almost no “overlapping” between tables. For instance, if we consider tables for \emptyset , then the table realized by a tuple (a_1, \dots, a_k) will not imply *anything* about the table realized by any non-identity permutation of the tuple (a_1, \dots, a_k) or any sub-tuple of (a_1, \dots, a_k) . And even if we are considering tables for $\{s\}$, the table realized by (a_1, \dots, a_k) will only imply something about the table realized by $(a_1, \dots, a_k, a_{k-1})$.

► **Definition 8.** Let \mathfrak{A} and \mathfrak{B} be models over the same vocabulary, and let \mathcal{F} be a subset of $\{I, s, E, \neg, C, \cap, \hat{\cap}\}$. Let $\bar{a} \in A^k$ and $\bar{b} \in B^k$, where $k \in \mathbb{Z}_+$. We say that \bar{a} and \bar{b} are similar with respect \mathcal{F} , if for every k -ary term $\mathcal{T} \in \text{GRA}(\mathcal{F})$ we have that $\bar{a} \in \llbracket \mathcal{T} \rrbracket_{\mathfrak{A}}$ if, and only if, $\bar{b} \in \llbracket \mathcal{T} \rrbracket_{\mathfrak{B}}$.

In what follows we will not mention the set \mathcal{F} explicitly, since it will always be clear from the context. For different subsets of $\{I, s, E, \neg, C, \cap, \hat{\cap}\}$ one can find explicit characterizations for when two tuples are similar using the notions of 1-types and tables. For example, if $\mathcal{F} = \{s, C, \neg, \cap\}$, then two tuples \bar{a} and \bar{b} are similar with respect to \mathcal{F} if and only if $tp_{\mathfrak{A}}(\bar{a}) = tp_{\mathfrak{B}}(\bar{b})$, $tp_{\mathfrak{A}}(a_{k-1}) = tp_{\mathfrak{B}}(b_{k-1})$ and $tp_{\mathfrak{A}}(a_k) = tp_{\mathfrak{B}}(b_k)$.

We will next introduce two Scott-normal forms for our logics. In the normal forms we will use the operator \cup which can be defined in a standard way in terms of \neg and \cap .

► **Definition 9.** Let $\mathcal{F} \subseteq \{I, s, E, C\}$.

■ We say that a term $\mathcal{T} \in \text{GRA}(\mathcal{F} \cup \{\neg, \cap, \exists\})$ is in normal form, if it has the following form

$$\bigcap_{1 \leq i \leq m_{\exists}} \exists \kappa_i \cap \bigcap_{1 \leq j \leq m_{\forall}} \forall \lambda_j \cap \bigcap_{1 \leq i \leq m_{\exists}} \forall^{n_i} (\neg \alpha_i^{\exists} \cup \exists \beta_i^{\exists}) \cap \bigcap_{1 \leq j \leq m_{\forall}} \forall^{n_j} (\neg \alpha_j^{\forall} \cup \forall \beta_j^{\forall}),$$

where κ_i , λ_j , α_i^{\exists} , β_i^{\exists} , α_j^{\forall} and β_j^{\forall} are terms of $\text{GRA}(\mathcal{F} \cup \{\neg, \cap\})$, and the terms κ_i and λ_j are unary. Here \forall is short-hand notation for $\neg \exists \neg$ and \forall^n stands for a sequence of \forall of length n .

■ We say that a term $\mathcal{T} \in \text{GRA}(\mathcal{F} \cup \{\neg, \cap, \hat{\cap}, \exists_1, \exists_0\})$ is in normal form, if it has the following form

$$\bigcap_{1 \leq i \leq m_{\exists}} \exists_0 \kappa_i \cap \bigcap_{1 \leq j \leq m_{\forall}} \forall_0 \lambda_j \cap \bigcap_{1 \leq i \leq m_{\exists}} \forall_0 (\neg \alpha_i^{\exists} \cup \exists_1 \beta_i^{\exists}) \cap \bigcap_{1 \leq j \leq m_{\forall}} \forall_0 (\neg \alpha_j^{\forall} \cup \forall_1 \beta_j^{\forall}),$$

where κ_i , λ_j , α_i^{\exists} , β_i^{\exists} , α_j^{\forall} and β_j^{\forall} are terms of $\text{GRA}(\mathcal{F} \cup \{\neg, \cap, \hat{\cap}\})$, and the terms κ_i and λ_j are unary. Here \forall_0 and \forall_1 are short-hand notations for $\neg \exists_0 \neg$ and $\neg \exists_1 \neg$ respectively.

In a rather standard fashion one can prove the following lemma.

► **Lemma 10.** Let $\mathcal{F} \subseteq \{I, s, E, C\}$.

1. There is a polynomial time nondeterministic procedure, taking as its input a term $\mathcal{T} \in \text{GRA}(\mathcal{F} \cup \{\neg, \cap, \exists\})$ and producing a term \mathcal{T}' in normal form (over extended signature), such that
 - if $\mathfrak{A} \models \mathcal{T}$, for some structure \mathfrak{A} , then there exists a run of the procedure which produces a term \mathcal{T}' in normal form so that $\mathfrak{A}' \models \mathcal{T}'$ for some expansion \mathfrak{A}' of \mathfrak{A} .
 - if the procedure has a run producing \mathcal{T}' and $\mathfrak{A} \models \mathcal{T}'$, for some \mathfrak{A} , then $\mathfrak{A} \models \mathcal{T}$.
2. There is a polynomial time nondeterministic procedure, which operates similarly as the above procedure with the exception that it takes as its input a term in $\mathcal{T} \in \text{GRA}(\mathcal{F} \cup \{\neg, \cap, \hat{\cap}, \exists_1, \exists_0\})$, and which satisfies the additional requirement that if \mathcal{T} does not contain the operator $\hat{\cap}$, then neither does any of the terms that this procedure produces.

To conclude this section, we will introduce some further notation and terminology which will be useful in the later sections of this paper. Consider a term \mathcal{T} in normal form. Subterms of \mathcal{T} that are of the form $\forall^{n_i} (\neg \alpha_i^{\exists} \cup \exists \beta_i^{\exists})$ or $\forall_0 (\neg \alpha_i^{\exists} \cup \exists_1 \beta_i^{\exists})$ are called **existential requirements** and we will denote them with \mathcal{T}_i^{\exists} . Similarly subterms of the form $\forall^{n_j} (\neg \alpha_j^{\forall} \cup \forall \beta_j^{\forall})$ or of the form $\forall_0 (\neg \alpha_j^{\forall} \cup \forall_1 \beta_j^{\forall})$ will be called **universal requirements** and we will denote them with \mathcal{T}_j^{\forall} . Consider a model \mathfrak{A} and an existential requirement \mathcal{T}_i^{\exists} . If

\mathcal{T}_i^{\exists} is of the form $\forall^{n_i}(\neg\alpha_i^{\exists} \cup \exists\beta_i^{\exists})$ and $\bar{a} \in \llbracket \alpha_i^{\exists} \rrbracket_{\mathfrak{A}}$, then an element $c \in A$ so that $\bar{a}c \in \llbracket \beta_i^{\exists} \rrbracket_{\mathfrak{A}}$ will be called a **witness** for \bar{a} and \mathcal{T}_i^{\exists} . Similarly, if \mathcal{T}_i^{\exists} is of the form $\forall_0(\neg\alpha_i^{\exists} \cup \exists_1\beta_i^{\exists})$ and $a \in \llbracket \alpha_i^{\exists} \rrbracket_{\mathfrak{A}}$, then a tuple $\bar{c} \in A^k$, where $k = ar(\beta_i^{\exists}) - 1$, is called a witness for a and \mathcal{T}_i^{\exists} .

5 Ordered logic with equality

In this section we will study the complexity of $\text{GRA}(E, \neg, \cap, \exists)$, i.e. ordered logic with equality. We will start by proving that this logic has a polynomially bounded model property, which means that each satisfiable term has a model of size at most polynomial with respect to the size of the term.

Before proceeding with the proof, we will first note that w.l.o.g. we can assume that if an element c is a witness for some existential requirement \mathcal{T}_i^{\exists} and a tuple (a_1, \dots, a_k) , then $a_k \neq c$. This follows from the observation that if \mathcal{T}_i^{\exists} is of the form $\forall^{n_i}(\neg\alpha_i^{\exists} \cup \exists\beta_i^{\exists})$ then it is equivalent with the following term $\forall^{n_i}(\neg(\alpha_i^{\exists} \cap \neg\exists E\beta_i^{\exists}) \cup \exists\beta_i^{\exists})$, where we can replace $\exists E\beta_i^{\exists}$ with $I\beta_i^{\exists}$.

► **Theorem 11.** *Let $\mathcal{T} \in \text{GRA}(I, E, \neg, \cap, \exists)$ and suppose that \mathcal{T} is satisfiable. Then \mathcal{T} has a model of size bounded polynomially in $|\mathcal{T}|$.*

Proof. Let $\mathcal{T} \in \text{GRA}(I, E, \neg, \cap, \exists)$ be a term in normal form. Let \mathfrak{A} be a model of \mathcal{T} . Without loss of generality we will assume that \mathfrak{A} contains at least two distinct elements. Our goal is to construct a bounded model $\mathfrak{B} \models \mathcal{T}$. As the domain of our model we will take the set

$$B = \{1, \dots, m\} \times \{0, 1\},$$

where $m = \max\{m_{\exists}^1, m_{\exists}\}$. To define the model, we just need to specify the tables for all the k -tuples of elements from B . This will be done inductively, and in such a way that the following condition is maintained: for every $\bar{b} \in B^k$ there exists $\bar{a} \in A^k$ so that \bar{b} is similar with \bar{a} . Maintaining this requirement will make sure that our model \mathfrak{B} will not violate any universal requirements.

We will start by defining the 1-types for all the elements of B . Since $\mathfrak{A} \models \bigcap_{1 \leq i \leq m_{\exists}^1} \exists \kappa_i$, for every $1 \leq i \leq m_{\exists}^1$ there exists $a_i \in A$ so that $tp_{\mathfrak{A}}(a_i) \models \kappa_i$. We will define that for every $(i, j) \in B$, $tp_{\mathfrak{B}}((i, j)) = tp_{\mathfrak{A}}(a_i)$. Suppose then that we have defined the tables for k -tuples and we wish to define the tables for $(k+1)$ -tuples. We will start by making sure that all the existential requirements are full-filled. So, let $1 \leq i \leq m_{\exists}$ and $\bar{b} \in B^k$ so that we have not assigned a witness for \bar{b} and \mathcal{T}_i^{\exists} . By construction we know that there exists $\bar{a} \in A^k$ which is similar to \bar{b} . Now there exists $a_k \neq c \in A$ so that $\bar{a}c \in \llbracket \beta_i^{\exists} \rrbracket$. If $b_k = (i', j)$, then we will use the element $d = (i, j + 1 \bmod 2)$ as a witness for \bar{b} by defining that $tp_{\mathfrak{B}}(\bar{b}d) = tp_{\mathfrak{A}}(\bar{a}c)$. Since we have reserved for every element $m_{\exists} \leq m$ distinct witnesses for the existential requirements, the process of providing witnesses can be done without conflicts.

Having provided witnesses for k -tuples, we will still need to do define the $(k+1)$ -tables for the remaining k -tables. So, let $\bar{b} \in B^k$ and $d \in B$ be elements so that the table of $\bar{b}d$ has not been defined. If $b_k = d$, then the table for $\bar{b}d$ is determined by the table for \bar{b} . Suppose then that $b_k \neq d$. Let $\bar{a} \in A^k$ be a k -tuple which is similar with \bar{b} . Pick an arbitrary $a_k \neq c \in A$ and define $tp_{\mathfrak{B}}(\bar{b}d) = tp_{\mathfrak{A}}(\bar{a}c)$. ◀

The above theorem can be used to show that if we assume that the underlying vocabulary to be bounded, i.e., there is a fixed constant bound on the maximum arity of relation symbols, then the complexity of the ordered logic is NP-complete.

► **Theorem 12.** *The satisfiability problem for $\text{GRA}(E, \neg, \cap, \exists)$ over bounded vocabularies is NP-complete.*

In the case where the vocabulary is not assumed to be bounded, the complexity of the ordered logic turns out to be PSPACE-complete. The complete proof can be found in the full version of this paper, but we will sketch the basic idea here. The PSPACE-hardness can be proved by reducing the satisfiability problem of modal logic over serial frames to that of $\text{GRA}(\neg, \cap, \exists)$. For the upper bound one can adapt the well-known algorithm of Ladner. The idea is to non-deterministically construct a model in a depth-first fashion by first guessing a set of 1-types of polynomial size (the domain of the model) and then guess tables for longer and longer tuples of elements.

► **Theorem 13.** *The satisfiability problem for $\text{GRA}(E, \neg, \cap, \exists)$ is PSPACE-complete.*

6 Further extensions of ordered logic

In this section we will study extensions of ordered logic which are obtained by adding either the swap or the one-dimensional intersection (or both) into its syntax. It turns out that we can deduce easily from the literature sharp lower bounds for the relevant fragments.

► **Proposition 14.** *Let \mathcal{F} be a set of relation operators that contains either $\{\neg, C, \cap, \exists\}$ or $\{s, \neg, \cap, \exists\}$. Now the satisfiability problem for $\text{GRA}(\mathcal{F})$ is NEXPTIME-hard.*

► **Remark 15.** We remark that the proof of the above proposition shows that the proposition holds even if we restrict attention to vocabularies which contain at most binary relation symbols. In particular, the satisfiability problems of $\text{GRA}(\neg, C, \cap, \exists_1, \exists_0)$ and $\text{GRA}(s, \neg, \cap, \exists_1, \exists_0)$ are also NEXPTIME-hard.

Now we will focus on proving the corresponding upper bounds on the complexities of $\text{GRA}(\neg, C, \cap, \exists)$ and $\text{GRA}(s, \neg, \cap, \exists)$ by showing that their least common extension $\text{GRA}(s, \neg, C, \cap, \exists)$ has the exponentially bounded model property. The core of the argument is the same as the proof of the exponential model property for FO^2 given in [3].

► **Theorem 16.** *Let $\mathcal{T} \in \text{GRA}(s, \neg, C, \cap, \exists)$ and suppose that \mathcal{T} is satisfiable. Then \mathcal{T} has a model of size bounded exponentially in $|\mathcal{T}|$.*

Proof. Let $\mathcal{T} \in \text{GRA}(s, \neg, C, \cap, \exists)$ be a term in normal form and let \mathfrak{A} be a model of \mathcal{T} . Our goal is to construct a bounded model \mathfrak{B} so that $\mathfrak{B} \models \mathcal{T}$. As the domain of the model \mathfrak{B} we will take the set

$$B := \{tp_{\mathfrak{A}}(a) \mid a \in A\} \times \{1, \dots, m\} \times \{0, 1, 2\},$$

where $m = \max\{m_{\exists}^1, m_{\exists}\}$. Clearly $|B| \leq 2^{O(|\mathcal{T}|)}$. Again, to construct the model, we will need to specify the tables for all the k -tuples of elements from B . We will follow the same strategy as in the proof of theorem 11, i.e. the tables will be specified inductively while maintaining the condition that for every $\bar{b} \in B^k$ for which $tp_{\mathfrak{B}}(\bar{b})$ has been specified, there exists $\bar{a} \in A^k$ which is similar to \bar{b} .

We will start with the 1-types. For every $b = (tp_{\mathfrak{A}}(a), i, j) \in B$ we define that $tp_{\mathfrak{B}}(b) := tp_{\mathfrak{A}}(a)$. Suppose then that we have defined the tables for k -tuples. We start defining the tables for $(k+1)$ -tuples by providing witnesses for all the relevant tuples. So, consider an existential requirement \mathcal{T}_i^{\exists} and a tuple $\bar{b} \in B^k$ so that $\bar{b} \in \llbracket \alpha_i \rrbracket_{\mathfrak{B}}$. Suppose that $b_k = (tp_{\mathfrak{A}}(a), i', j)$. By construction there exists a tuple $\bar{a} \in A^k$ so that \bar{b} and \bar{a} are similar. Thus $\bar{a} \in \llbracket \alpha_i \rrbracket_{\mathfrak{A}}$. Since

$\mathfrak{A} \models \mathcal{T}_i^{\exists}$, there exists an element $c \in A$ which is a witness for \bar{a} and \mathcal{T}_i^{\exists} . We will use the element $d = (tp_{\mathfrak{A}}(c), i, j+1 \bmod 3) \in B$ as a witness for \bar{b} by defining that $tp_{\mathfrak{B}}(\bar{b}d) := tp_{\mathfrak{A}}(\bar{a}c)$ and $tp_{\mathfrak{B}}((b_1, \dots, b_{k-1}, d, b_k)) := tp_{\mathfrak{A}}((a_1, \dots, a_{k-1}, c, a_k))$.

Before moving forward, let us argue that our method of assigning witnesses does not produce conflicts. Consider a tuple $\bar{b} = (b_1, \dots, b_k) \in B^k$ and $d \in B$ so that we used d as a witness for \bar{b} and some existential requirement \mathcal{T}_i^{\exists} . We will argue that the table for the tuple (\bar{b}, d) was not defined in two different ways. First we note that we have reserved distinct elements for each of the existential requirements, and thus we used d as a witness for \bar{b} only for the existential requirement \mathcal{T}_i^{\exists} . We then note that since we are assigning witnesses for tuples in a “cyclic” manner, we will not use b_k as a witness for the tuple (b_1, \dots, b_{k-1}, d) . Since these cases are the only possible ways that we might have defined the table of the tuple $\bar{b}d$ in two different ways, we conclude that it is only defined once.

We will now assign tables for the remaining $(k+1)$ -tuples. So, consider a tuple $\bar{b} \in B^k$ and $d = (tp_{\mathfrak{A}}(c), i, j) \in B$ so that we have not defined the table for the tuple $\bar{b}d$. By construction there exists a tuple $\bar{a} \in A^k$ which is similar to \bar{b} . Let $c \in A$ be an element that realizes the 1-type of d (and which is not necessarily distinct from a_k). Now we define that $tp_{\mathfrak{B}}(\bar{b}d) := tp_{\mathfrak{A}}(\bar{a}c)$ and that $tp_{\mathfrak{B}}((b_1, \dots, b_{k-1}, d, b_k)) = tp_{\mathfrak{A}}((a_1, \dots, a_{k-1}, c, a_k))$. ◀

► **Corollary 17.** *The satisfiability problem for $\text{GRA}(s, \neg, C, \cap, \exists)$ is NEXPTIME-complete.*

We will conclude this section by considering $\text{GRA}(E, \neg, C, \cap, \exists)$ and $\text{GRA}(s, E, \neg, C, \cap, \exists)$. An easy modification in the argument of theorem 11 yields a bounded model property for the first logic.

► **Theorem 18.** *Let $\mathcal{T} \in \text{GRA}(E, \neg, C, \cap, \exists)$ and suppose that \mathcal{T} is satisfiable. Then \mathcal{T} has a model of size bounded exponentially in $|\mathcal{T}|$.*

Proof. Let \mathcal{T} be a term in normal form and assume that \mathfrak{A} is a model of \mathcal{T} . If $K = \{tp_{\mathfrak{A}}(a) \mid a \text{ is a king}\}$, then one can take as the domain of the bounded model \mathfrak{B} the set

$$B := K \cup (\{tp_{\mathfrak{A}}(a) \mid a \text{ is not a king}\} \times \{1, \dots, m\} \times \{0, 1\}),$$

where $m = \max\{m_{\exists}^1, m_{\exists}\}$. One can now adapt the proof of theorem 11 to obtain a model \mathfrak{B} of \mathcal{T} with domain B . ◀

► **Corollary 19.** *The satisfiability problem for $\text{GRA}(E, \neg, C, \cap, \exists)$ is NEXPTIME-complete.*

The logic $\text{GRA}(s, E, \neg, C, \cap, \exists)$ turns out to be more tricky. We have not been able to verify whether this logic is undecidable, but we can show that it does not have the finite model property, see the full version of this paper.

7 One-dimensional ordered logics

In this section we consider logics that are obtained from the ordered logic and the fluted logic by imposing the restriction of one-dimensionality. We will first show that the satisfiability problem of the one-dimensional fluted logic, which has been extended with the operators s and E , is NEXPTIME-complete. As usual, we will prove this by showing that the logic has the bounded model property. The proof is heavily influenced by similar model constructions performed in [9] and [8], which were based on the classical construction of [3].

► **Theorem 20.** *Let $\mathcal{T} \in \text{GRA}(s, E, \neg, \dot{\cap}, \exists_1, \exists_0)$ and suppose that \mathcal{T} is satisfiable. Then \mathcal{T} has a model of size bounded exponentially in $|\mathcal{T}|$.*

62:12 Ordered Fragments of First-Order Logic

Proof. Let $\mathcal{T} \in \text{GRA}(s, E, \neg, \hat{\cap}, \exists_1, \exists_0)$ be a term in normal form and suppose that \mathcal{T} is satisfiable. Let us fix an arbitrary model \mathfrak{A} of \mathcal{T} , which we will use to construct a bounded model \mathfrak{B} for \mathcal{T} . Let $K \subseteq A$ denote the set of kings of A . For each existential requirement \mathcal{T}_i^\exists of \mathcal{T} and $k \in \llbracket \alpha_i^\exists \rrbracket_{\mathfrak{A}} \cap K$, we will pick some witness \bar{c} . Let C denote the resulting set. Next, we let P denote the set of non-royal 1-types realized by elements of \mathfrak{A} . Fix some function $f : P \rightarrow A$ with the property that $tp_{\mathfrak{A}}(f(\pi)) = \pi$, for every $\pi \in P$. For every existential requirement \mathcal{T}_i^\exists and $\pi \in P$ so that $\pi \models \alpha_i^\exists$, we will pick some witness $\bar{c}^{\pi,i}$. Let $W_{\pi,i}$ denote the set of elements occurring in $\bar{c}^{\pi,i}$ that are not kings.

As the domain of the bounded model \mathfrak{B} we will then take the following set

$$B = C \cup \bigcup_{\pi,i,j} W_{\pi,i,j},$$

where π ranges over P , i ranges over $\{1, \dots, m\}$, where $m = \max\{m_\exists^1, m_\exists\}$, and j ranges over $\{0, 1, 2\}$. The sets $W_{\pi,i,j}$ are pairwise disjoint copies of the sets $W_{\pi,i}$. Clearly $|B| \leq 2^{O(|\mathcal{T}|)}$. We will make $\mathfrak{B} \upharpoonright C$ isomorphic with $\mathfrak{A} \upharpoonright C$. Furthermore, we will make each of the structures $\mathfrak{B} \upharpoonright (K \cup W_{\pi,i,j})$ isomorphic with the corresponding structures $\mathfrak{A} \upharpoonright (K \cup W_{\pi,i})$.

We will then provide witnesses for elements of B . Since we have already provided witnesses for kings, we need to only provide witnesses for non-royal elements of the court and for elements in $(B \setminus C)$. We will start with the non-royal elements of the court. Consider an existential requirement \mathcal{T}_i^\exists and let $b \in (C \setminus K) \cap \llbracket \alpha_i^\exists \rrbracket_{\mathfrak{A}}$. If there exists a witness for b and \mathcal{T}_i^\exists in C , then nothing needs to be done. So suppose that there does not exist a witness for b and \mathcal{T}_i^\exists in C . If π is the 1-type of b in \mathfrak{B} , then we know that there exists a witness \bar{c} for $f(\pi)$ and \mathcal{T}_i^\exists . We have now two cases.

Suppose first that the length of \bar{c} is one, i.e. $\bar{c} = c$, for some $c \in A$. If $c = a$, then b is already a witness for itself in \mathfrak{B} . If $c \neq a$, then we define $tp_{\mathfrak{B}}(b, d) = tp_{\mathfrak{B}}(a, c)$ and $tp_{\mathfrak{B}}(d, b) = tp_{\mathfrak{B}}(c, a)$, where d denotes the single element of $W_{\pi,i,0}$ (note that d can't be a king, since otherwise b and \mathcal{T}_i^\exists would have had a witness in C). Suppose then that the length of \bar{c} is $k > 1$. If $\bar{d} \in (W_{\pi,i,0} \cup K)^k$ denotes the corresponding witness, then we define $tp_{\mathfrak{B}}(b\bar{d}) = tp_{\mathfrak{A}}(a\bar{c})$ and $tp_{\mathfrak{B}}(bd_1, \dots, d_k, d_{k-1}) = tp_{\mathfrak{A}}(ac_1, \dots, c_k, c_{k-1})$. Note that since b does not occur in \bar{d} and \bar{d} contains at least one non-royal element, the above definitions do not lead into any conflicts with the structure that we have assigned for $\mathfrak{B} \upharpoonright C$.

Thus we have managed to provide witnesses for elements in $C \setminus K$. To provide witnesses for elements of $(B \setminus C)$, we can do roughly the same as above with the exception that instead of $W_{\pi,i,0}$, we will use - assuming that $b \in W_{\pi',i',j}$ - the set $W_{\pi,i,j+1 \bmod 3}$. Let us then briefly argue that the above procedure for producing witnesses can be executed without conflicts. First we note that we do not face any conflicts when assigning witnesses for some b and \mathcal{T}_i^\exists and then for b and $\mathcal{T}_{i'}^\exists$, where $i \neq i'$, since for every j the sets $W_{\pi,i,j}$ and $W_{\pi,i',j}$ are disjoint. Secondly we note that we do not face any conflicts when assigning witnesses for some b and \mathcal{T}_i^\exists and then for $b \neq b'$ and \mathcal{T}_i^\exists , since in the first case we assign a table for the tuple $b\bar{d}$ and in the second case for $b'\bar{d}$, and neither of these tables imply anything about the other table. Finally we note that since we are assigning witnesses in a cyclic manner, if we use \bar{d} as a witness for $b \notin C$ and \mathcal{T}_i^\exists , then we are never using any tuple containing b as a witness for any of the elements in \bar{d} .

To complete the structure, for every k we need to define the tables for tuples $\bar{b} \in B^k$. We can do this inductively with respect to k as follows. Suppose first that there exists distinct elements $b, b' \in B$ so that we have not assigned table for the pair (b, b') . Now we choose a pair of distinct elements $a, a' \in A$ with the same 1-types as b and b' , and then define $tp_{\mathfrak{B}}(b, b') = tp_{\mathfrak{A}}(a, a')$ and $tp_{\mathfrak{B}}(b', b) = tp_{\mathfrak{A}}(a', a)$. Note that such elements a, a' exist even if the elements b, b' would have the same 1-types, since at least one of them is not a king.

Suppose then that we have defined the tables for every $\bar{d} \in B^k$. Let $b \in B$ and $\bar{d} \in B^k$ be so that we have not defined the table for the tuple (b, \bar{d}) . By construction there exists $\bar{c} \in A^k$ which is similar with \bar{d} . Let $a \in A$ be an arbitrary element which has the same 1-type as b . We then define $tp_{\mathfrak{B}}(b, \bar{d}) = tp_{\mathfrak{A}}(a, \bar{c})$ and $tp_{\mathfrak{B}}(b, d_1, \dots, d_k, d_{k-1}) = tp_{\mathfrak{A}}(a, c_1, \dots, c_k, c_{k-1})$. Continuing this way it is clear that we can define tables for all the tuples of B^k in such a way that we do not violate any of the universal requirements. ◀

► **Corollary 21.** *The satisfiability problem for $\text{GRA}(s, E, \neg, \dot{\cap}, \exists_1, \exists_0)$ is NEXPTIME-complete.*

We will conclude this section by considering the one-dimensional ordered logic with equality, which is the logic $\text{GRA}(E, \neg, \cap, \exists_1, \exists_0)$. Perhaps unsurprisingly, the satisfiability problem for this logic is NP-complete.

► **Theorem 22.** *The satisfiability problem for $\text{GRA}(E, \neg, \cap, \exists_1, \exists_0)$ is NP-complete.*

8 Conclusions

In this paper we have studied systematically how the complexities of various ordered fragments of first-order logic change if we modify slightly the underlying syntax. The general picture that emerges is that even if we relax only slightly the restrictions on the syntax, the complexity of the logic can increase drastically. On the other hand, we have seen that adding the further restriction of one-dimensionality on the logics can greatly decrease the complexity of the logic.

There are several directions in which the work conducted in this paper can be continued. Perhaps the most immediate technical problem is whether the logic $\text{GRA}(s, E, \neg, C, \cap, \exists)$ is decidable. As we have seen, this logic does not have the finite model property, and thus we don't expect that traditional model building techniques can be used to prove that it is decidable. On the other hand, we have not been able to prove that this logic is undecidable using standard tiling arguments.

References

- 1 Bartosz Bednarczyk. Exploiting forwardness: Satisfiability and query-entailment in forward guarded fragment. In *JELIA*, pages 179–193. Springer, 2021.
- 2 Erich Grädel. On the restraining power of guards. *The Journal of Symbolic Logic*, 64(4):1719–1742, 1999.
- 3 Erich Grädel, Phokion G Kolaitis, and Moshe Y Vardi. On the decision problem for two-variable first-order logic. *Bulletin of symbolic logic*, 3(1):53–69, 1997.
- 4 Andreas Herzig. A new decidable fragment of first order logic. In *3rd Logical Biennial Summer School and Conference in Honour of SC Kleene*, 1990.
- 5 Ullrich Hustadt, Renate A Schmidt, and Lilia Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1(251-276):3, 2004.
- 6 Reijo Jaakkola and Antti Kuusisto. Algebraic classifications for fragments of first-order logic and beyond. *arXiv Preprint*, 2020. [arXiv:2005.01184v1](https://arxiv.org/abs/2005.01184v1).
- 7 Reijo Jaakkola and Antti Kuusisto. Algebraic classifications for fragments of first-order logic and beyond. *arXiv Preprint*, 2021. [arXiv:2005.01184v2](https://arxiv.org/abs/2005.01184v2).
- 8 Emanuel Kieronski. One-dimensional guarded fragments. *arXiv Preprint*, 2019. [arXiv:1904.04572](https://arxiv.org/abs/1904.04572).
- 9 Emanuel Kieroński and Antti Kuusisto. Complexity and expressivity of uniform one-dimensional fragment with equality. In *International Symposium on Mathematical Foundations of Computer Science*, pages 365–376. Springer, 2014.

62:14 Ordered Fragments of First-Order Logic

- 10 Antti Kuusisto. On games and computation. *arXiv Preprint*, 2019. [arXiv:1910.14603](#).
- 11 Martin Otto. Two variable first-order logic over ordered domains. *The Journal of Symbolic Logic*, 66(2):685–702, 2001.
- 12 Ian Pratt-Hartmann. Fluted logic with counting. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 13 Ian Pratt-Hartmann, Wiesław Szwał, and Lidia Tendera. The fluted fragment revisited. *The Journal of Symbolic Logic*, 84(3):1020–1048, 2019.
- 14 Ian Pratt-Hartmann and Lidia Tendera. The fluted fragment with transitive relations. *arXiv Preprint*, 2020. [arXiv:2006.11169](#).
- 15 Willard V Quine. Variables explained away. *Proceedings of the american philosophical society*, 104(3):343–347, 1960.
- 16 WV Quine. Predicate functors revisited. *The Journal of Symbolic Logic*, 46(3):649–652, 1981.
- 17 Wiesław Szwał and Lidia Tendera. On the decision problem for the guarded fragment with transitivity. In *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science*, pages 147–156. IEEE, 2001.

The Simplest Non-Regular Deterministic Context-Free Language

Petr Jančar  

Dept. of Computer Science, Faculty of Science, Palacký University Olomouc, Czech Republic

Jiří Šíma 

Institute of Computer Science of the Czech Academy of Sciences, Prague, Czech Republic

Abstract

We introduce a new notion of \mathcal{C} -simple problems for a class \mathcal{C} of decision problems (i.e. languages), w.r.t. a particular reduction. A problem is \mathcal{C} -simple if it can be reduced to each problem in \mathcal{C} . This can be viewed as a conceptual counterpart to \mathcal{C} -hard problems to which all problems in \mathcal{C} reduce. Our concrete example is the class of *non-regular* deterministic context-free languages (DCFL'), with a truth-table reduction by Mealy machines. The main technical result is a proof that the DCFL' language $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ is DCFL'-simple, and can be thus viewed as one of the simplest languages in the class DCFL', in a precise sense. The notion of DCFL'-simple languages is nontrivial: e.g., the language $L_R = \{wcw^R \mid w \in \{a, b\}^*\}$ is not DCFL'-simple.

By describing an application in the area of neural networks (elaborated in another paper), we demonstrate that \mathcal{C} -simple problems under suitable reductions can provide a tool for expanding the lower-bound results known for single problems to the whole classes of problems.

2012 ACM Subject Classification Theory of computation \rightarrow Grammars and context-free languages; Theory of computation \rightarrow Problems, reductions and completeness; Theory of computation \rightarrow Transducers

Keywords and phrases deterministic context-free language, truth-table reduction, Mealy automaton, pushdown automaton

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.63

Related Version *Full Version*: <https://arxiv.org/abs/2102.10416>

Funding Presented research has been partially supported by the Czech Science Foundation, grant GA19-05704S, and by the institutional support RVO: 67985807 (J. Šíma).

Acknowledgements J. Šíma also thanks Martin Plátek for his intensive collaboration at the first stages of this research.

1 Introduction

We introduce a new notion of \mathcal{C} -simple problems for a class \mathcal{C} of decision problems (i.e. languages). A problem is \mathcal{C} -simple if it can be reduced to each problem in \mathcal{C} ; if this problem is, moreover, in \mathcal{C} , it can be viewed as a simplest problem in \mathcal{C} . The \mathcal{C} -simple problems are thus a conceptual counterpart to the common \mathcal{C} -hard problems (like, e.g., NP-hard problems) to which conversely any problem in \mathcal{C} reduces. These definitions (of \mathcal{C} -simple and \mathcal{C} -hard problems) are parametrized by a chosen reduction that does not have a higher computational complexity than the class \mathcal{C} itself. Therefore, it may be said that if a \mathcal{C} -hard problem has a (computationally) “easy” solution, then each problem in \mathcal{C} has an “easy” solution. On the other hand, if we prove that a \mathcal{C} -simple problem is not “easy”, in particular that it cannot be solved by machines of a type \mathcal{M} that can implement the respective reduction, then all problems in \mathcal{C} are not “easy”, that is, are not solvable by \mathcal{M} ; this extends a lower-bound result for one problem to the whole class of problems.



© Petr Jančar and Jiří Šíma;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 63; pp. 63:1–63:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper, we consider \mathcal{C} to be the class of non-regular deterministic context-free languages, which we denote by DCFL' ; we thus have $\text{DCFL}' = \text{DCFL} \setminus \text{REG}$ (where REG denotes the class of regular languages). We use a truth-table reduction by Mealy machines (which is motivated below). Hence a DCFL' -simple problem is a language $L_0 \subseteq \Sigma^*$ (over an alphabet Σ) that can be reduced to each DCFL' language $L \subseteq \Delta^*$ by a Mealy machine \mathcal{A} with an oracle L , denoted \mathcal{A}^L . More precisely, we have a finite-state transducer \mathcal{A} that transforms a given input word $w \in \Sigma^*$ to a word $\mathcal{A}(w) \in \Delta^*$ (a query prefix), and each state q of \mathcal{A} is associated with a finite tuple $\sigma_q = (s_{q,1}, s_{q,2}, \dots, s_{q,r_q})$ of r_q words from Δ^* (query suffixes), and with a truth table $f_q : \{0,1\}^{r_q} \rightarrow \{0,1\}$. The oracle-machine \mathcal{A}^L behaves like \mathcal{A} , hence it reads an input word w (translating it to $\mathcal{A}(w)$) by which it enters a state q , and then submits r_q queries, i.e. the words $\mathcal{A}(w) \cdot s_{q,i}$ for all $i \in \{1, 2, \dots, r_q\}$, to the oracle that for each $i \in \{1, 2, \dots, r_q\}$ decides whether or not $\mathcal{A}(w) \cdot s_{q,i}$ is in L (or, equivalently, whether or not $\mathcal{A}(w)$ belongs to the quotient $L/s_{q,i} = \{v \in \Delta^* \mid v \cdot s_{q,i} \in L\}$); the oracle-answers are then aggregated by the truth table f_q , which decides whether or not $w \in L_0$.

This truth-table reduction by Mealy machines induces a preorder on the class of languages; we denote this preorder by \leq_{tt}^A , using the superscript “A” to stress that our truth-table reduction is realized by simple *automata*, not by general Turing machines. The main technical result of this paper is that the DCFL' language $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ (over the binary alphabet $\{0,1\}$) is DCFL' -simple, since $L_{\#} \leq_{tt}^A L$ for each language L in DCFL' . The class DCFLS of DCFL' -simple languages comprises REG and is a strict subclass of DCFL ; e.g., the DCFL' language $L_R = \{w c w^R \mid w \in \{a,b\}^*\}$ over the alphabet $\{a,b,c\}$ proves to be not DCFL' -simple. The closure properties of DCFLS are similar to that of DCFL as the class DCFLS is closed under complement and intersection with regular languages, while being not closed under concatenation, intersection, and union.

The above definition of DCFL' -simple problems has originally been motivated by the analysis of the computational power of neural network (NN) models which is known to depend on the (descriptive) complexity of their weight parameters [9, 12]. The so-called analog neuron hierarchy [10] of binary-state NNs with increasing number of α extra analog-state neurons, denoted as αANN for $\alpha \geq 0$, has been introduced for studying NNs with realistic weights between integers (finite automata) and rational numbers (Turing machines). We use the notation αANN also for the class of languages accepted by αANNs , which can clearly be distinguished by the context. The separation $1\text{ANN} \subsetneq 2\text{ANN}$ has been witnessed by the DCFL' language $L_{\#} \in 2\text{ANN} \setminus 1\text{ANN}$. The proof of $L_{\#} \notin 1\text{ANN}$ is rather technical (based on the Bolzano-Weierstrass theorem) which could hardly be generalized to other DCFL' languages, while it was conjectured that $L \notin 1\text{ANN}$ for all DCFL' languages L , that is, $\text{DCFL}' \subseteq (2\text{ANN} \setminus 1\text{ANN})$ (implying $1\text{ANN} \cap \text{DCFL} = 0\text{ANN} = \text{REG}$). An idea how to prove this conjecture is to show that $L_{\#}$ is in some sense the simplest problem in the class DCFL' , namely, to reduce $L_{\#}$ to any DCFL' language L by using a reduction that can be carried out by 1ANNs , which are at least as powerful as finite automata. If the composition of a 1ANN that carries out the reduction of $L_{\#}$ to L with a hypothetical 1ANN accepting L can be realized by another 1ANN , which would thus accept $L_{\#}$, we get that no 1ANN accepting L can exist, since $L_{\#}$ has been proven not to be accepted by 1ANNs .

The idea why $L_{\#}$ should serve as the simplest language in the class DCFL' comes from the fact that any reduced context-free grammar G generating a non-regular language $L \subseteq \Delta^*$ is *self-embedding* [4, Theorem 4.10]. This means that there is a so-called self-embedding nonterminal A admitting the derivation $A \Rightarrow^* xAy$ for some non-empty strings $x, y \in \Delta^+$. Since G is reduced, there are strings $v, w, z \in \Delta^*$ such that $S \Rightarrow^* vAz$ and $A \Rightarrow^* w$ where S is the start nonterminal in G , which implies $S \Rightarrow^* vx^mwy^mz \in L$ for every $m \geq 0$. It is thus

straightforward to suggest to reduce an input word $0^m 1^n \in \{0, 1\}^*$ where $m, n \geq 1$, to the string $vx^m wy^n z \in \Delta^*$ (while the inputs outside $0^+ 1^+$ are mapped onto some fixed string outside L) since $0^m 1^n \in L_{\#}$ entails $vx^m wy^n z \in L$.

However, the suggested (one-one) reduction from $L_{\#}$ to L is not consistent because $vx^m wy^n z \in L$ does not necessarily imply $0^m 1^n \in L_{\#}$. For example, consider the DCFL' language $L_1 = \{0^m 1^n \mid 1 \leq m \leq n\}$ over the binary alphabet $\Delta = \{0, 1\}$ for which there are no words $v, x, w, y, z \in \Delta^*$ such that $vx^m wy^n z \in L_1$ would ensure $m = n$. Nevertheless, we can pick two inputs $0^m 1^{n-1}$ and $0^m 1^n$ instead of one, that is, $x = 0$, $y = 1$, and $v = w = z = \varepsilon$ (ε denoting the empty string), which satisfy $0^m 1^n \in L_{\#}$ iff $m = n$ iff $vx^m wy^{n-1} z \notin L_1$ and $vx^m wy^n z \in L_1$.

It turns out that this can be generalized to any DCFL' language. Namely, we prove in this paper that for each DCFL' language $L \subseteq \Delta^*$, over an alphabet Δ , there are words $v, x, w, y, z \in \Delta^+$ and a language $L' \in \{L, \bar{L}\}$, where $\bar{L} = \Delta^* \setminus L$ is the complement of L , such that $0^m 1^n \in L_{\#}$ iff $vx^m wy^{n-1} z \notin L'$ and $vx^m wy^n z \in L'$. In fact, we even show that either for all $m, n \geq 0$ we have $vx^m wy^n z \in L'$ iff $m = n$, or for all $m, n \geq 0$ we have $vx^m wy^n z \in L'$ iff $m \leq n$. We note that this technical result seems interesting on its own since in the class DCFL it substantially strengthens the known result for context-free languages (CFL) [2, Theorem 2.10] that for any CFL' language $L \subseteq \Delta^*$ (where $\text{CFL}' = \text{CFL} \setminus \text{REG}$) there is a so-called non-degenerated iterative pair $(v, x, w, y, z) \in (\Delta^*)^5$ with non-empty xy , satisfying $vx^m wy^m z \in L$ for all $m \geq 0$ and $vx^m wy^n z \notin L$ for some $m \neq n$.

Hence the inconsistent many-one (in fact, one-one) reduction from $L_{\#}$ with one query to the oracle L is replaced by a truth-table reduction, that is, by a special Turing reduction in which all its finitely many (in our case two) oracle queries are presented at the same time and there is a Boolean function (a truth table) which, when given the answers to the queries, produces the final answer of the reduction. This truth-table reduction from $L_{\#}$ to L can be implemented by a deterministic finite-state transducer (a Mealy machine) \mathcal{A} with the oracle L : It transforms the input $0^m 1^n$ where $m, n \geq 1$ (the inputs outside $0^+ 1^+$ are rejected), to the output $vx^m wy^{n-1} \in \Delta^+$ and carries out two queries to L that arise by concatenation of this output with two fixed suffixes z and yz ; hence the queries are $vx^m wy^{n-1} z \stackrel{?}{\in} L$ and $vx^m wy^n z \stackrel{?}{\in} L$. The truth table is defined so that the input $0^m 1^n$ is accepted by \mathcal{A}^L iff the two answers to these queries are distinct and at same time, the first answer is negative in the case $L' = L$, and positive in the case $L' = \bar{L}$, which is equivalent to $0^m 1^n \in L_{\#}$.

It follows that the DCFL' language $L_{\#}$ is DCFL'-simple under the truth-table reduction by Mealy machines. Since this reduction can be implemented by 1ANNs, we achieve the desired stronger separation $\text{DCFL}' \subseteq (\text{2ANN} \setminus \text{1ANN})$ in the analog neuron hierarchy [10, 11]. This result constitutes a non-trivial application of the proposed concept of DCFL'-simple problem. Moreover, if we could generalize the result to (nondeterministic) CFL, e.g. by proving that some DCFL' language is CFL'-simple, which would imply that $L_{\#}$ is CFL'-simple by the transitivity of reduction, then we would achieve that even the intersection of CFL' and 1ANN is empty. We note the interesting fact that $L_{\#}$ cannot be CSL'-simple (under our reduction), since 1ANN accepts some context-sensitive languages outside CFL [10].

In general, if we show that some \mathcal{C} -simple problem under a given reduction cannot be computed by a computational model \mathcal{M} that implements this reduction, then all problems in the class \mathcal{C} are not solvable by \mathcal{M} either. The notion of \mathcal{C} -simple problems can thus be useful for expanding known (e.g. technical) lower-bound results for individual problems to the whole classes of problems at once, as it was the case of the DCFL'-simple problem $L_{\#} \notin \text{1ANN}$, expanding to $\text{DCFL}' \cap \text{1ANN} = \emptyset$. It seems worthwhile to explore if looking for \mathcal{C} -simple problems in other complexity classes \mathcal{C} could provide effective tools for strengthening known lower bounds.

We remark that the hardest context-free language by Greibach [3] can be viewed as CFL-hard under a special type of our reduction \leq_{tt}^A . Related line of study concerns the types of reductions used in finite or pushdown automata with oracle. For example, non-deterministic finite automata with oracle complying with many-one restriction have been applied to establishing oracle hierarchies over the context-free languages [8]. For the same purpose, oracle pushdown automata have been used for many-one, truth-table, and Turing reducibilities, respectively, inducing the underlying definitions also to oracle nondeterministic finite automata [14]. In addition, nondeterministic finite automata whose oracle queries are completed by the prefix of an input word that has been read so far and the remaining suffix, have been employed in defining a polynomial-size oracle hierarchy [1].

In the preliminary study [13], some considerations about the simplest DCFL' language have appeared, yet without formal definitions of DCFL'-simple problems, that included only sketches of incomplete proofs of weaker results based on the representation of DCFL by so-called deterministic monotonic restarting automata [6], which have initiated investigations of non-regularity degrees in DCFL [7].

In this paper we achieve a complete argument for $L_{\#}$ to be a DCFL'-simple problem, within the framework of deterministic pushdown automata (DPDA) by using some ideas on regularity of pushdown processes from [5]. We now give an informal overview of the proof. Given a DPDA \mathcal{M} recognizing a non-regular language $L \subseteq \Delta^*$, it is easy to realize that some computations of \mathcal{M} (from the initial configuration) must be reaching configurations where the stack is arbitrarily large while it can be (almost) erased afterwards. Hence the existence of words $v, x, w, y, z \in \Delta^+$ such that $vx^mwy^mz \in L$ for all $m \geq 0$ is obvious. However, we aim to guarantee that for all m, n the equality $m = n$ holds if, and only if, $vx^mwy^{n-1}z \notin L'$ and $vx^mwy^n z \in L'$, where L' is either the language L or its complement. This is not so straightforward but it is confirmed by our detailed analysis (in Section 3). We study the computation of \mathcal{M} on an infinite word $a_1a_2a_3 \dots$ that visits infinitely many pairwise non-equivalent configurations. We use a natural congruence property of language equivalence on the set of configurations, and avoid some tedious technical details by a particular use of Ramsey's theorem. This allows us to extract the required tuple $v, x, w, y, z \in \Delta^+$ from the mentioned infinite computation. We note that determinism of \mathcal{M} is essential in the presented proof; we leave open if it can be relaxed to show that $L_{\#}$ is even CFL'-simple.

The rest of the paper is organized as follows. In Section 2 we recall basic definitions and notation regarding DPDA and Mealy machines, introduce the novel concept of DCFL'-simple problems under truth-table reduction by Mealy machines and show some simple properties of the class DCFLS of DCFL'-simple problems. In Section 3 we present the proof of the main technical result which shows that $L_{\#}$ is DCFL'-simple. Finally, we summarize the results and list some open problems in Section 4.

2 DCFL'-Simple Problem Under Truth-Table Mealy Reduction

In this section we define the truth-table reduction by Mealy machines, introduce the notion of DCFL'-simple problems, show their basic properties, and formulate the main technical result (Theorem 1). But first we recall standard definitions of pushdown automata.

A *pushdown automaton (PDA)* is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, R, q_0, X_0, F)$ where Q is a finite set of states including the start state $q_0 \in Q$ and the set $F \subseteq Q$ of accepting states, while the finite sets $\Sigma \neq \emptyset$ and $\Gamma \neq \emptyset$ represent the input and stack alphabets, respectively, with the initial stack symbol $X_0 \in \Gamma$. In addition, the set R contains finitely many transition rules

$pX \xrightarrow{a} q\gamma$ with the meaning that \mathcal{M} in state $p \in Q$, on the input $a \in \Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ (recall that ε denotes the empty string), and with $X \in \Gamma$ as the topmost stack symbol may read a , change the state to $q \in Q$, and pop X , replacing it by pushing $\gamma \in \Gamma^*$.

By a *configuration* of \mathcal{M} we mean $p\alpha \in Q \times \Gamma^*$, and we define relations \xrightarrow{a} for $a \in \Sigma_\varepsilon$ on $Q \times \Gamma^*$: each rule $pX \xrightarrow{a} q\gamma$ in R induces $pX\alpha \xrightarrow{a} q\gamma\alpha$ for all $\alpha \in \Gamma^*$; these relations are naturally extended to \xrightarrow{w} for $w \in \Sigma^*$. For a configuration $p\alpha$ we define $\mathcal{L}(p\alpha) = \{w \in \Sigma^* \mid p\alpha \xrightarrow{w} q\beta \text{ for some } q \in F \text{ and } \beta \in \Gamma^*\}$, and $\mathcal{L}(\mathcal{M}) = \mathcal{L}(q_0X_0)$ is the language accepted by \mathcal{M} . A PDA \mathcal{M} is *deterministic* (a DPDA) if there is at most one rule $pX \xrightarrow{a} ..$ for each tuple $p \in Q$, $X \in \Gamma$, $a \in \Sigma_\varepsilon$; moreover, if there is a rule $pX \xrightarrow{\varepsilon} ..$, then there is no rule $pX \xrightarrow{a} ..$ for $a \in \Sigma$. We also use the standard assumption that all ε -steps are popping, that is, in each rule $pX \xrightarrow{\varepsilon} q\gamma$ in R we have $\gamma = \varepsilon$.

The languages accepted by (deterministic) pushdown automata constitute the class of (*deterministic*) *context-free languages*; the classes are denoted by DCFL and CFL, respectively, whereas $\text{DCFL}' = \text{DCFL} \setminus \text{REG}$.

In the following theorem we formulate the main technical result: any language in DCFL' includes a certain “projection” of the language $L_\# = \{0^n1^n \mid n \geq 1\}$, which means that $L_\#$ is in some sense the simplest language in the class DCFL' . The theorem, whose proof will be presented in Section 3, thus provides an interesting property of DCFL' .

► **Theorem 1.** *Let $L \subseteq \Delta^*$ be a non-regular deterministic context-free language over an alphabet Δ . There exist non-empty words $v, x, w, y, z \in \Delta^+$ and a language $L' \in \{L, \bar{L}\}$ (where $\bar{L} = \Delta^* \setminus L$ is the complement of L) such that*

- *either for all $m, n \geq 0$ we have $vx^mwy^n z \in L'$ iff $m = n$,*
- *or for all $m, n \geq 0$ we have $vx^mwy^n z \in L'$ iff $m \leq n$;*

this entails that for all $m \geq 0$ and $n > 0$ we have

$$(vx^mwy^{n-1}z \notin L' \text{ and } vx^mwy^n z \in L') \quad \text{iff} \quad m = n. \quad (1)$$

In order to formalize the DCFL' -simple problems, we now define a *Mealy machine* \mathcal{A} with an *oracle*: it is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, \delta, \lambda, q_0, \{(\sigma_q, f_q) \mid q \in Q\})$ where Q is a finite set of states including the start state $q_0 \in Q$, and the finite sets $\Sigma \neq \emptyset$ and $\Delta \neq \emptyset$ represent the input and output (oracle) alphabets, respectively. Moreover, $\delta : Q \times \Sigma \rightarrow Q$ is a (partial) state-transition function which extends to input strings as $\delta : Q \times \Sigma^* \rightarrow Q$ where $\delta(q, \varepsilon) = q$ for every $q \in Q$, while $\delta(q, wa) = \delta(\delta(q, w), a)$ for all $q \in Q$, $w \in \Sigma^*$, $a \in \Sigma$. Similarly, $\lambda : Q \times \Sigma \rightarrow \Delta^*$ is an output function which extends to input strings as $\lambda : Q \times \Sigma^* \rightarrow \Delta^*$ where $\lambda(q, \varepsilon) = \varepsilon$ for all $q \in Q$, and $\lambda(q, wa) = \lambda(q, w) \cdot \lambda(\delta(q, w), a)$ for all $q \in Q$, $w \in \Sigma^*$, $a \in \Sigma$. In addition, for each $q \in Q$, the tuple $\sigma_q = (s_{q,1}, s_{q,2}, \dots, s_{q,r_q})$ of strings in Δ^* contains r_q query suffixes, while $f_q : \{0, 1\}^{r_q} \rightarrow \{0, 1\}$ is a truth table that aggregates the answers to the r_q oracle queries.

The above Mealy machine \mathcal{A} starts in the start state q_0 and operates as a deterministic finite-state transducer that transforms an input word $w \in \Sigma^*$ to the output string $\mathcal{A}(w) = \lambda(q_0, w) \in \Delta^*$ written to a so-called oracle tape. The oracle tape is a semi-infinite, write-only tape which is empty at the beginning and its contents are only extended in the course of computation by appending the strings to the right. Namely, given a current state $q \in Q$ and an input symbol $a \in \Sigma$, the machine \mathcal{A} moves to the next state $\delta(q, a) \in Q$ and writes the string $\lambda(q, a) \in \Delta^*$ to the oracle tape, if $\delta(q, a)$ is defined; otherwise \mathcal{A} rejects the input. After reading the whole input word $w \in \Sigma^*$, the machine \mathcal{A} is in the state $p = \delta(q_0, w) \in Q$, while the oracle tape contains the output $\mathcal{A}(w) = \lambda(q_0, w) \in \Delta^*$.

Finally, the Mealy machine \mathcal{A} , equipped with an oracle $L \subseteq \Delta^*$, in this case denoted \mathcal{A}^L , queries the oracle whether $\mathcal{A}(w)$ belongs to the (right) quotient $L/s_{p,i} = \{u \in \Delta^* \mid u \cdot s_{p,i} \in L\}$, for each suffix $s_{p,i}$ in σ_p , and the answers are aggregated by the truth table f_p . Thus, the oracle Mealy machine \mathcal{A}^L accepts the input word $w \in \Sigma^*$ iff

$$f_p \left(\chi_{L/s_{p,1}}(\mathcal{A}(w)), \chi_{L/s_{p,2}}(\mathcal{A}(w)), \dots, \chi_{L/s_{p,r_p}}(\mathcal{A}(w)) \right) = 1$$

where $p = \delta(q_0, w)$ and $\chi_{L/s_{p,i}} : \Delta^* \rightarrow \{0, 1\}$ is the characteristic function of $L/s_{p,i}$, that is, $\chi_{L/s_{p,i}}(u) = 1$ if $u \cdot s_{p,i} \in L$, and $\chi_{L/s_{p,i}}(u) = 0$ if $u \cdot s_{p,i} \notin L$. The language accepted by the machine \mathcal{A}^L is defined as $\mathcal{L}(\mathcal{A}^L) = \{w \in \Sigma^* \mid w \text{ is accepted by } \mathcal{A}^L\}$.¹

We say that $L_1 \subseteq \Sigma^*$ is *truth-table reducible* to $L_2 \subseteq \Delta^*$ by a Mealy machine, which is denoted as $L_1 \leq_{tt}^A L_2$, if $L_1 = \mathcal{L}(\mathcal{A}^{L_2})$ for some Mealy machine \mathcal{A} running with the oracle L_2 . The following lemma shows that we can chain these reductions together since the relation \leq_{tt}^A is a preorder.

► **Lemma 2.** *The relation \leq_{tt}^A is reflexive and transitive.*

Proof. The relation \leq_{tt}^A is reflexive since $L = \mathcal{L}(\mathcal{A}^L) \subseteq \Sigma^*$ for the oracle Mealy machine $\mathcal{A}^L = (\{q\}, \Sigma, \Sigma, \delta, \lambda, q, \{(\sigma_q, f_q)\})$ where $\delta(q, a) = q$ and $\lambda(q, a) = a$ for every $a \in \Sigma$, $\sigma_q = (\varepsilon)$, and f_q is the identity.

Now we show that the relation \leq_{tt}^A is transitive. Let $L_1 \leq_{tt}^A L_2$ and $L_2 \leq_{tt}^A L_3$ which means $L_1 = \mathcal{L}(\mathcal{A}_1^{L_2}) \subseteq \Sigma^*$ and $L_2 = \mathcal{L}(\mathcal{A}_2^{L_3}) \subseteq \Delta^*$ for some oracle Mealy machines $\mathcal{A}_1^{L_2} = (Q_1, \Sigma, \Delta, \delta_1, \lambda_1, q_0^1, \{(\pi_q, g_q) \mid q \in Q_1\})$ and $\mathcal{A}_2^{L_3} = (Q_2, \Delta, \Theta, \delta_2, \lambda_2, q_0^2, \{(\varrho_q, h_q) \mid q \in Q_2\})$, respectively. We will construct the oracle Mealy machine $\mathcal{A}^{L_3} = (Q, \Sigma, \Theta, \delta, \lambda, q_0, \{(\sigma_q, f_q) \mid q \in Q\})$ such that $L_1 = \mathcal{L}(\mathcal{A}^{L_3}) \subseteq \Sigma^*$ which implies the transitivity $L_1 \leq_{tt}^A L_3$. We define $Q = Q_1 \times Q_2$ with $q_0 = (q_0^1, q_0^2)$, $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, \lambda_1(q_1, a)))$ and $\lambda((q_1, q_2), a) = \lambda_2(q_2, \lambda_1(q_1, a))$ for every $(q_1, q_2) \in Q$ and $a \in \Sigma$, which ensures $\mathcal{A}(w) = \lambda(q_0, w) = \lambda_2(q_0^2, \lambda_1(q_0^1, w)) = \mathcal{A}_2(\mathcal{A}_1(w)) \in \Theta^*$ for every $w \in \Sigma^*$. For each state $p = (p_1, p_2) \in Q$ in \mathcal{A} , we define the tuple of query suffixes from Θ^* ,

$$\sigma_p = (\lambda_2(p_2, s_{p_1,i}) \cdot s_{p_2(i),j} \mid i = 1, \dots, r_{p_1}, j = 1, \dots, r_{p_2(i)})$$

where $\pi_{p_1} = (s_{p_1,1}, s_{p_1,2}, \dots, s_{p_1,r_{p_1}}) \in \Delta^{r_{p_1}}$ and $\varrho_{p_2(i)} = (s_{p_2(i),1}, s_{p_2(i),2}, \dots, s_{p_2(i),r_{p_2(i)}}) \in \Theta^{r_{p_2(i)}}$ are the query suffixes associated with $p_1 \in Q_1$ and $p_2(i) = \delta_2(p_2, s_{p_1,i}) \in Q_2$ for $i \in \{1, \dots, r_{p_1}\}$, respectively, and the truth table $f_p = g_{p_1}(h_{p_2(1)}, \dots, h_{p_2(r_{p_1})})$ aggregates the answers to the corresponding oracle queries, which ensures $L_1 = \mathcal{L}(\mathcal{A}^{L_3}) \subseteq \Sigma^*$. ◀

We say that $L_0 \subseteq \Sigma^*$ is *DCFL'-simple* if $L_0 \leq_{tt}^A L$ for every non-regular deterministic context-free language $L \subseteq \Delta^*$. We show that Theorem 1 entails that the DCFL' language $L_\#$ is DCFL'-simple. In addition, we denote by DCFLS the class of DCFL'-simple problems and formulate its basic properties.

► **Corollary 3** (of Theorem 1). *The non-regular deterministic context-free language $L_\# = \{0^n 1^n \mid n \geq 1\}$ is DCFL'-simple.*

Proof. Let $L \subseteq \Delta^*$ be any DCFL' language. According to Theorem 1, there are $v, x, w, y, z \in \Delta^+$ and $L' \in \{L, \bar{L}\}$ such that condition (1) holds for L' . We define the Mealy machine $\mathcal{A}^L = (\{q_0, q_1, q_2\}, \{0, 1\}, \Delta, \delta, \lambda, q_0, \{(\sigma_q, f_q) \mid q \in Q\})$ with the oracle L , as $\delta(q_0, 0) = \delta(q_1, 0) = q_1$, $\delta(q_1, 1) = \delta(q_2, 1) = q_2$, $\lambda(q_0, 0) = vx$, $\lambda(q_1, 0) = x$, $\lambda(q_1, 1) = w$, $\lambda(q_2, 1) = y$, $\sigma_{q_0} = \sigma_{q_1} = ()$

¹ Note that the described protocol works also for non-prefix-free languages since for any input prefix that has been read so far, the output value from the truth table determines whether the oracle Mealy machine is in an “accepting” state, deciding about this prefix analogously as a deterministic finite automaton. The truth-table reduction only requires that the given oracle answers do not influence further computation when subsequent input symbols are read.

($r_{q_0} = r_{q_1} = 0$), $\sigma_{q_2} = (z, yz)$ ($r_{q_2} = 2$), $f_{q_0} = f_{q_1} = 0$, $f_{q_2}(0,0) = f_{q_2}(1,1) = 0$, and $f_{q_2}(1,0) = 1 - f_{q_2}(0,1)$ where $f_{q_2}(0,1) = 1$ iff $L' = L$. It is easy to verify that $L_{\#} = \mathcal{L}(\mathcal{A}^L)$, which implies $L_{\#} \leq_{tt}^A L$. Hence, $L_{\#}$ is DCFL'-simple. ◀

► **Proposition 4.**

1. $REG \subsetneq DCFLS$.
2. $DCFLS \subsetneq DCFL$, and $L_R = \{wcw^R \mid w \in \{a, b\}^*\} \in DCFL \setminus DCFLS$.
3. The class $DCFLS$ is closed under complement and intersection with regular languages.
4. The class $DCFLS$ is not closed under concatenation, intersection and union.

Proof (Sketch).

1. For any regular language L , consider a Mealy machine $\mathcal{A}^{L_{\#}}$ with the DCFL'-simple oracle $L_{\#}$, that simulates a deterministic finite automaton recognizing L , while its constant truth tables produce 1 iff associated with the accept states. Hence, $L \leq_{tt}^A L_{\#}$ which means L is DCFL'-simple according to Lemma 2 and Corollary 3 which also implies $REG \neq DCFLS$.
2. We first observe that $DCFLS \subseteq DCFL$. Let $L \in DCFLS$ be any DCFL'-simple language which ensures $L \leq_{tt}^A L_{\#}$ by an oracle Mealy machine $\mathcal{A}^{L_{\#}}$. The machine $\mathcal{A}^{L_{\#}}$ can be simulated by a DPDA \mathcal{M} which extends a suitable DPDA $\mathcal{M}_{\#}$ (e.g. with no ε -transitions) accepting $L_{\#} = \mathcal{L}(\mathcal{M}_{\#})$, so that the finite control of \mathcal{M} implements the finite-state transducer \mathcal{A} whose output is presented online as an input to $\mathcal{M}_{\#}$. Moreover, for each state q of \mathcal{A} , the finite control of \mathcal{M} evaluates the truth table f_q which aggregates the answers to the queries with r_q suffixes associated with q , by inspecting at most constant number of topmost stack symbols. Hence $L = \mathcal{L}(\mathcal{M}) \in DCFL$.

In order to show that $DCFLS \neq DCFL$, we prove that the DCFL $L_R = \{wcw^R \mid w \in \{a, b\}^*\}$ over the alphabet $\{a, b, c\}$ is not DCFL'-simple. For the sake of contradiction, suppose that $L_R \leq_{tt}^A L_{\#}$ by a Mealy machine $\mathcal{A}^{L_{\#}} = (Q, \{a, b, c\}, \{0, 1\}, \delta, \lambda, q_0, \{(\sigma_q, f_q) \mid q \in Q\})$ with the oracle $L_{\#} = \{0^n 1^n \mid n \geq 1\}$, which means $L_R = \mathcal{L}(\mathcal{A}^{L_{\#}})$. Consider all the 2^k possible prefixes $w \in \{a, b\}^k$ of inputs presented to $\mathcal{A}^{L_{\#}}$ that have the length $|w| = k$. These strings can bring $\mathcal{A}^{L_{\#}}$ into a finite number $|\{\delta(q_0, w) \mid w \in \{a, b\}^k\}| \leq |Q|$ of distinct states while the length $|\lambda(q_0, w)|$ of outputs written to the oracle tape is bounded by $O(k)$. For $\lambda(q_0, w)$ outside 0^*1^* , the acceptance of words wu where $u \in \{a, b, c\}^*$, depends only on the truth values $f_q(0, \dots, 0)$ associated with the states q from the finite set Q , due to $\lambda(q_0, wu) \notin L_{\#}/s$ for any $s \in \{0, 1\}^*$. On the other hand, the number of distinct outputs $\lambda(q_0, w)$ in 0^*1^* is bounded by $O(k)$. This means that for a sufficiently large $k \geq 1$, there must be two distinct prefixes $w_1, w_2 \in \{a, b\}^k$ such that $\delta(q_0, w_1) = \delta(q_0, w_2)$ and $\lambda(q_0, w_1) = \lambda(q_0, w_2)$ in 0^*1^* , which results in the contradiction $w_1 c w_2^R \in \mathcal{L}(\mathcal{A}^{L_{\#}}) \setminus L_R$.

3. The class $DCFLS$ is closed under complement since the truth tables can be negated. Furthermore, any oracle Mealy machine can be modified so that it simulates another given finite automaton in parallel and is forced to reject if this automaton rejects, which shows $DCFLS$ to be closed under intersection with regular languages.
4. Observe that $R = \{1\}^*$, $L_1 = \{0^m 1^m 0^n \mid m, n \geq 1\}$, $L_2 = \{0^m 1^n 0^n \mid m, n \geq 1\}$, and $L_3 = L_1 \cup (\{1\} \cdot L_2)$ are DCFL'-simple while $R \cdot L_3 \notin DCFL$, $L_1 \cap L_2 \notin CFL$, and $L_1 \cup L_2 \notin DCFL$ are not DCFL'-simple according to 2. ◀

3 Proof of the Main Result (Theorem 1)

Theorem 1 follows from the (more specific) next lemma that we prove in this section. (See Appendix for an informal overview with figures.)

By \mathbb{N} we denote the set $\{0, 1, 2, \dots\}$, and by $[i, j]$ the set $\{i, i+1, \dots, j\}$ (for $i, j \in \mathbb{N}$).

► **Lemma 5.** *Let $\mathcal{M} = (Q, \Sigma, \Gamma, R, p_0, X_0, F)$ be a DPDA where $L = \mathcal{L}(\mathcal{M}) = \mathcal{L}(p_0X_0)$ is non-regular (hence L belongs to DCF L'). There are $v \in \Sigma^*$, $x, w, y, z \in \Sigma^+$, $p, q \in Q$, $X \in \Gamma$, $\gamma \in \Gamma^+$, $\delta \in \Gamma^*$ such that the following four conditions hold:*

1. $p_0X_0 \xrightarrow{v} pX\delta$ and $pX \xrightarrow{x} pX\gamma$, which entails the infinite (stack increasing) computation

$$p_0X_0 \xrightarrow{v} pX\delta \xrightarrow{x} pX\gamma\delta \xrightarrow{x} pX\gamma\gamma\delta \xrightarrow{x} pX\gamma\gamma\gamma\delta \xrightarrow{x} \dots; \quad (2)$$

2. $pX \xrightarrow{w} q$;
3. $q\gamma \xrightarrow{y} q$, hence $q\gamma^\ell\delta' \xrightarrow{y^\ell} q\delta'$ for all $\ell \in \mathbb{N}$ and $\delta' \in \Gamma^*$;
4. one of the following cases is valid (depending on whether $z \in \mathcal{L}(q\delta)$ or $z \notin \mathcal{L}(q\delta)$):
 - a. $\mathcal{L}(q\gamma^k\delta) \ni y^\ell z$ iff $k = \ell$ (for all $k, \ell \in \mathbb{N}$), or $\mathcal{L}(q\gamma^k\delta) \ni y^\ell z$ iff $k \leq \ell$ (for all $k, \ell \in \mathbb{N}$);
 - b. $\mathcal{L}(q\gamma^k\delta) \ni y^\ell z$ iff $k \neq \ell$ (for all $k, \ell \in \mathbb{N}$), or $\mathcal{L}(q\gamma^k\delta) \ni y^\ell z$ iff $k > \ell$ (for all $k, \ell \in \mathbb{N}$).

We note that $p_0X_0 \xrightarrow{v} pX\delta \xrightarrow{x^m} pX\gamma^m\delta \xrightarrow{w} q\gamma^m\delta \xrightarrow{y^m} q\delta$ (for each $m \in \mathbb{N}$); hence $vx^mwy^mz \in L$ iff $z \in \mathcal{L}(q\delta)$ (since z is nonempty). Theorem 1 indeed follows from the lemma: there is $L' \in \{L, \bar{L}\}$ such that either $vx^mwy^mz \in L'$ iff $m = n$ (for all $m, n \in \mathbb{N}$), or $vx^mwy^mz \in L'$ iff $m \leq n$ (for all $m, n \in \mathbb{N}$). (In Theorem 1 we also stated that v is nonempty. If $v = \varepsilon$ here, then we simply take vx and yz as the new v, z , respectively.)

Proof of Lemma 5

In the rest of this section we provide a proof of Lemma 5, assuming a fixed DPDA $\mathcal{M} = (Q, \Sigma, \Gamma, R, p_0, X_0, F)$ where $L = \mathcal{L}(p_0X_0)$ is non-regular. The proof structure is visible from the auxiliary claims that we state and prove on the way.

Convention. W.l.o.g. we assume that \mathcal{M} always reads the whole input $u \in \Sigma^*$ from p_0X_0 . This can be accomplished in the standard way, by adding a special bottom-of-stack symbol \perp and a (non-accepting) fail-state. (Each empty-stack configuration $q\varepsilon$ becomes $q\perp$, and each originally stuck computation enters the fail-state where it loops. We also recall that all ε -steps are popping, and thus infinite ε -sequences are impossible.) Hence for any infinite word $a_1a_2a_3 \dots$ in Σ^ω there is a unique infinite computation of \mathcal{M} starting in p_0X_0 ; it stepwise reads the whole infinite word $a_1a_2a_3 \dots$.

The *left quotient of L by $u \in \Sigma^*$* is the set $u \setminus L = \{v \in \Sigma^* \mid uv \in L\}$; concatenation has priority over \setminus , hence $u_1u_2 \setminus L = (u_1u_2) \setminus L$. (The next claim is valid for any non-regular L .)

► **Claim 6.** We can fix an infinite word $a_1a_2a_3 \dots$ in Σ^ω ($a_i \in \Sigma$) such that $a_1a_2 \dots a_i \setminus L \neq a_1a_2 \dots a_j \setminus L$ for all $i \neq j$.

Proof. Let us consider the labelled transition system $\mathcal{T} = (\text{LQ}(L), \Sigma, (\xrightarrow{a})_{a \in \Sigma})$ where $\text{LQ}(L) = \{u \setminus L \mid u \in \Sigma^*\}$ and $\xrightarrow{a} = \{(L', a \setminus L') \mid L' \in \text{LQ}(L)\}$. (We recall that $L' = u \setminus L$ entails $a \setminus L' = ua \setminus L$.) Since L is non-regular, the set of states reachable from $L = \varepsilon \setminus L$ in \mathcal{T} is infinite. The out-degree of states in \mathcal{T} is finite (in fact, bounded by $|\Sigma|$), hence an application of König's lemma yields an infinite *acyclic* path $L \xrightarrow{a_1} L_1 \xrightarrow{a_2} L_2 \xrightarrow{a_3} \dots$. ◁

We call a *configuration* $p\alpha$ of \mathcal{M} *unstable* if $\alpha = Y\beta$ and R contains a rule $pY \xrightarrow{\varepsilon} q$ (we recall that ε -steps are only popping); otherwise $p\alpha$ is *stable*. Since \mathcal{M} is a *deterministic* PDA, for each unstable $p\alpha$ we can soundly define the *stable successor of $p\alpha$* as the unique stable

configuration $p'\alpha'$ where $p\alpha \xrightarrow{\varepsilon} p'\alpha'$ (α' being a suffix of α). If the path $p\alpha \xrightarrow{\varepsilon} p'\alpha'$ does not go via an accepting state (in F), then $\mathcal{L}(p\alpha) = \mathcal{L}(p'\alpha')$; otherwise $\mathcal{L}(p\alpha) = \{\varepsilon\} \cup \mathcal{L}(p'\alpha')$. (We note that the configurations in the computation (2) that start with pX are necessarily stable: since we have $pX \xrightarrow{x} pX\gamma$ for $x \in \Sigma^+$, we cannot have $pX \xrightarrow{\varepsilon} p'$.)

▷ **Claim 7.** Each configuration is visited at most twice by

$$\text{the computation of } \mathcal{M} \text{ from } p_0X_0 \text{ on } a_1a_2a_3 \cdots \text{ that is fixed by Claim 6.} \quad (3)$$

Proof. The computation (3) is infinite, stepwise reading the whole word $a_1a_2a_3 \cdots$, and it can be presented as

$$r_0\gamma_0 \xrightarrow{a_1} r_1\gamma_1 \xrightarrow{a_2} r_2\gamma_2 \xrightarrow{a_3} \cdots \quad (\text{for } r_0\gamma_0 = p_0X_0)$$

where each $r_i\gamma_i$ is stable; each segment $r_i\gamma_i \xrightarrow{a_{i+1}} r_{i+1}\gamma_{i+1}$ starts with a (visible) a_{i+1} -step that is followed by a (maybe empty) sequence of (popping) ε -steps via unstable configurations. Since such an ε -sequence might go through an accepting state, we can have $r_i\gamma_i = r_j\gamma_j$ for $i \neq j$ though $a_1a_2 \cdots a_i \setminus L \neq a_1a_2 \cdots a_j \setminus L$; in this case L contains precisely one of the words $a_1a_2 \cdots a_i$ and $a_1a_2 \cdots a_j$, and the languages $a_1a_2 \cdots a_i \setminus L$ and $a_1a_2 \cdots a_j \setminus L$ differ just on ε . Nevertheless, this reasoning entails that we cannot have $r_i\gamma_i = r_j\gamma_j = r_\ell\gamma_\ell$ for pairwise different i, j, ℓ .

Since each segment $r_i\gamma_i \xrightarrow{a_{i+1}} r_{i+1}\gamma_{i+1}$ visits any unstable configuration at most once and $r_{i+1}\gamma_{i+1}$ is the stable successor for all unstable configurations in the segment, we deduce that also each unstable configuration can be visited at most twice in the computation (3). ◁

▷ **Claim 8.** The computation (3) on $a_1a_2a_3 \cdots$ can be “stair-factorized”, that is, written

$$p_0X_0 \xrightarrow{v_0} p_1X_1\alpha_1 \xrightarrow{v_1} p_2X_2\alpha_2\alpha_1 \xrightarrow{v_2} p_3X_3\alpha_3\alpha_2\alpha_1 \xrightarrow{v_3} \cdots \quad (4)$$

so that for each $i \in \mathbb{N}$ we have $v_i \in \Sigma^+$ and $p_iX_i \xrightarrow{v_i} p_{i+1}X_{i+1}\alpha_{i+1}$ where α_{i+1} is a nonempty suffix of the right-hand side of a rule in R (i.e., a nonempty suffix of γ in a rule $pX \xrightarrow{a} q\gamma$).

Proof. We consider the computation (3), and call a stable configuration $pX\beta$ a *level*, with *position* $i \in \mathbb{N}$, if $p_0X_0 \xrightarrow{a_1 \cdots a_i} pX\beta$ and all configurations visited by the computation $pX\beta \xrightarrow{a_{i+1}a_{i+2} \cdots}$ after $pX\beta$ have the stack longer than $|X\beta|$; each level $pX\beta$ has thus a unique position which we denote $\text{POS}(pX\beta)$. Since each configuration is visited at most twice in (3), and the set of configurations with a fixed length is finite, we get that the set of levels is infinite, with elements $p'_0X'_0, p_1X_1\beta_1, p_2X_2\beta_2, \dots$ where $0 \leq \text{POS}(p'_0X'_0) < \text{POS}(p_1X_1\beta_1) < \text{POS}(p_2X_2\beta_2) < \dots$. The computation (3) can be thus presented as

$$p_0X_0 \xrightarrow{v'_0} p'_0X'_0 \xrightarrow{v''_0} p_1X_1\beta_1 \xrightarrow{v_1} p_2X_2\beta_2 \xrightarrow{v_2} p_3X_3\beta_3 \xrightarrow{v_3} \cdots$$

where $|v'_0| = \text{POS}(p'_0X'_0)$, and $|v_0v_1 \cdots v_{j-1}| = \text{POS}(p_jX_j\beta_j)$ for $j \geq 1$, putting $v_0 = v'_0v''_0$.

Each segment $pX\beta \xrightarrow{v} p'X'\beta'$ between two neighbouring levels can be obviously written as $pX\beta \xrightarrow{a} q\gamma_1\gamma_2\beta \xrightarrow{v'} p'X'\gamma_2\beta$ where $pX \xrightarrow{a} q\gamma_1\gamma_2$ is a rule in R , both γ_1 and γ_2 are nonempty, $v = av'$, and $q\gamma_1 \xrightarrow{v'} p'X'$. Hence the validity of the claim is clear. ◁

We define the natural *equivalence relation* \sim on the set of configurations of \mathcal{M} : we put $p\alpha \sim q\beta$ if $\mathcal{L}(p\alpha) = \mathcal{L}(q\beta)$.

We fix the presentation (4), calling $p_iX_i\alpha_i\alpha_{i-1} \cdots \alpha_1$ the *level-configurations* (for all $i \in \mathbb{N}$). Since we have $\mathcal{L}(p_iX_i\alpha_i\alpha_{i-1} \cdots \alpha_1) \setminus \{\varepsilon\} = (v_0v_1 \cdots v_{i-1} \setminus L) \setminus \{\varepsilon\}$, there cannot be three level-configurations in the same \sim -class (i.e., in the same equivalence class w.r.t. \sim).

63:10 The Simplest Non-Regular Deterministic Context-Free Language

Hence any infinite set of level-configurations represents infinitely many \sim -classes. Now we show a congruence-property that might enable to shorten a level-configuration while its \sim -class is preserved. We use the notation $\text{DS}(p\alpha)$ (the “down-states” of $p\alpha$), putting

$$\text{DS}(p\alpha) = \{q \mid p\alpha \xrightarrow{u} q \text{ for some } u \in \Sigma^*\}.$$

▷ **Claim 9.** If $q\gamma \sim q\gamma'$ for each $q \in \text{DS}(p\beta)$, then $p\beta\gamma \sim p\beta\gamma'$.

Proof. Let us consider $u \in \Sigma^*$. If $u \in \mathcal{L}(p\beta)$, then $u \in \mathcal{L}(p\beta\mu)$ for all $\mu \in \Gamma^*$. If $u \notin \mathcal{L}(p\beta)$ and there is no prefix u' of u such that $p\beta \xrightarrow{u'} q$, then $u \notin \mathcal{L}(p\beta\mu)$ for all $\mu \in \Gamma^*$. If $u \notin \mathcal{L}(p\beta)$ and $u = u'u''$ where $pX\beta \xrightarrow{u'} q$ (necessarily for some $q \in \text{DS}(pX\beta)$), then $u \in \mathcal{L}(p\beta\mu)$ iff $u'' \in \mathcal{L}(q\mu)$. Hence the claim is clear. ◁

The next claim is an immediate corollary.

▷ **Claim 10.** Any computation $p_0X_0 \xrightarrow{w_1} pX\beta_1 \xrightarrow{w_2} pX\beta_2\beta_1 \xrightarrow{w_3} p'X'\beta_3\beta_2\beta_1$ where $pX \xrightarrow{w_2} pX\beta_2$ ($w_2 \in \Sigma^+$), $pX \xrightarrow{w_3} p'X'\beta_3$, and $q\beta_2\beta_1 \sim q\beta_1$ for each $q \in \text{DS}(p'X'\beta_3)$ can be shortened to $p_0X_0 \xrightarrow{w_1} pX\beta_1 \xrightarrow{w_3} p'X'\beta_3\beta_1$ where $p'X'\beta_3\beta_1 \sim p'X'\beta_3\beta_2\beta_1$.

The i -th level-configuration in (4) is reached by the computation $p_0X_0 \xrightarrow{v_0v_1\cdots v_{i-1}} p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1$. It can happen that there are j_1, j_2 , $0 \leq j_1 < j_2 \leq i$ such that $p_{j_1}X_{j_1} = p_{j_2}X_{j_2}$ and $q\alpha_{j_2}\alpha_{j_2-1}\cdots\alpha_1 \sim q\alpha_{j_1}\alpha_{j_1-1}\cdots\alpha_1$ for all $q \in \text{DS}(p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_{j_2+1})$. In this case we can shorten the computation as in Claim 10, where $v_{j_1}v_{j_1+1}\cdots v_{j_2-1}$ corresponds to the omitted w_2 . The resulting shorter computation might be possible to be repeatedly shortened further (if it can be presented so that the conditions of Claim 10 are satisfied). Now for each $i \geq 1$ we fix a (stair-factorized) computation

$$p_{i,0}X_{i,0} \xrightarrow{v_{i,0}} p_{i,1}X_{i,1}\alpha_{i,1} \xrightarrow{v_{i,1}} p_{i,2}X_{i,2}\alpha_{i,2}\alpha_{i,1} \cdots \xrightarrow{v_{i,n_i-1}} p_{i,n_i}X_{i,n_i}\alpha_{i,n_i}\alpha_{i,n_i-1} \cdots \alpha_{i,1} \quad (5)$$

that has arisen by a maximal sequence of the above shortenings of the prefix

$$p_0X_0 \xrightarrow{v_0v_1\cdots v_{i-1}} p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1 \text{ of (4).}$$

Hence $p_{i,0}X_{i,0} = p_0X_0$, $p_{i,n_i}X_{i,n_i} = p_iX_i$, $\alpha_{i,n_i}, \alpha_{i,n_i-1}, \dots, \alpha_{i,1}$ is a subsequence of $\alpha_i, \alpha_{i-1}, \dots, \alpha_1$, and $p_{i,n_i}X_{i,n_i}\alpha_{i,n_i}\alpha_{i,n_i-1}\cdots\alpha_{i,1} \sim p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1$.

▷ **Claim 11.** For each $\ell \in \mathbb{N}$ there is i such that $n_i > \ell$ (where n_i is from (5)).

Proof. As already discussed, the set of level-configurations represents infinitely many \sim -classes. The last configurations of computations (5) represent the same infinite set of \sim -classes, and their lengths thus cannot be bounded; since the lengths of all $\alpha_{i,j}$ are bounded (they are shorter than the longest right-hand sides of the rules in R), the claim is clear. ◁

Now we come to a crucial claim in our proof of Lemma 5. Besides the notation $\text{DS}(p\alpha)$ we also introduce $\text{ES}(p\alpha)$ (the by- ε -reached down-states of $p\alpha$), by putting

$$\text{ES}(p\alpha) = \{q \mid p\alpha \xrightarrow{\varepsilon} q\}.$$

Hence $\text{ES}(p\alpha) \subseteq \text{DS}(p\alpha)$, and $|\text{ES}(p\alpha)| \leq 1$ (due to the determinism of the DPDA \mathcal{M}).

We recall that $p\alpha \sim q\beta$ means $\mathcal{L}(p\alpha) = \mathcal{L}(q\beta)$. To handle the special case of the empty word ε , we also define a (much) coarser equivalence \sim_0 : we put $p\alpha \sim_0 q\beta$ if ε either belongs to both $\mathcal{L}(p\alpha)$ and $\mathcal{L}(q\beta)$, or belongs to none of them.

The next claim is rather technical but it captures some straightforward combinatorial observations that are handled by a simple use of Ramsey's theorem. Informally speaking, if n_i in the final configuration in (5) is sufficiently large, then we can find a convenient pumping segment in this configuration. (All this should be easily understandable after reading the informal overview with figures in Appendix.)

▷ **Claim 12.** There is a constant $B \in \mathbb{N}$ determined by the DPDA \mathcal{M} such that for all $i \in \mathbb{N}$ where $n_i > B$ the final configuration in (5) can be written as

$$p_{i,n_i} X_{i,n_i} \alpha_{i,n_i} \alpha_{i,n_i-1} \cdots \alpha_{i,1} = \bar{p}\bar{X}\beta\gamma\delta$$

where the following conditions hold:

1. $\gamma = \alpha_{i,j} \alpha_{i,j-1} \cdots \alpha_{i,j'+1}$ where $n_i \geq j > j' \geq n_i - B$ and $p_{i,j} X_{i,j} = p_{i,j'} X_{i,j'}$ (and $\beta = \alpha_{i,n_i} \alpha_{i,n_i-1} \cdots \alpha_{i,j+1}$, $\delta = \alpha_{i,j'} \alpha_{i,j'-1} \cdots \alpha_{i,1}$);
2. the sets $\text{DS}(\bar{p}\bar{X}\beta)$ and $\text{DS}(\bar{p}\bar{X}\beta\gamma)$ are equal, further being denoted by \bar{Q} ;
3. for each $q \in \bar{Q}$, if $\text{ES}(q\gamma) = \{q'\}$, then $\text{ES}(q'\gamma) = \{q'\}$ (and $q' \in \bar{Q}$);
4. each $q' \in \bar{Q}$ belongs to $\text{DS}(q\gamma)$ for some self-containing $q \in \bar{Q}$, where $q \in \bar{Q}$ is *self-containing* if $q \in \text{DS}(q\gamma)$;
5. there is a state $q' \in \bar{Q}$ for which $q'\gamma\delta \not\sim q'\delta$ and $q'\gamma\delta \sim_0 q'\delta$.

Proof. We fix some i with n_i larger than a constant B determined by \mathcal{M} as described below (there are such i by Claim 11). For convenience we put $p_{i,n_i} X_{i,n_i} = \bar{p}\bar{X}$, $n_i = n$, and $\alpha_{i,j} = \bar{\alpha}_j$, hence the final configuration in (5) is $p_{i,n_i} X_{i,n_i} \alpha_{i,n_i} \alpha_{i,n_i-1} \cdots \alpha_{i,1} = \bar{p}\bar{X}\bar{\alpha}_n \bar{\alpha}_{n-1} \cdots \bar{\alpha}_1$. We view the $n+1$ prefixes

$$\bar{p}\bar{X}, \bar{p}\bar{X}\bar{\alpha}_n, \bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}, \bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}\bar{\alpha}_{n-2}, \dots, \bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1} \cdots \bar{\alpha}_1$$

as the vertices of a complete graph with coloured edges.

For $\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1} \cdots \bar{\alpha}_1 = \bar{p}\bar{X}\mu\nu\rho$, where $\mu = \bar{\alpha}_n\bar{\alpha}_{n-1} \cdots \bar{\alpha}_{j+1}$, $\nu = \bar{\alpha}_j\bar{\alpha}_{j-1} \cdots \bar{\alpha}_{j'+1}$, and $\rho = \bar{\alpha}_{j'}\bar{\alpha}_{j'-1} \cdots \bar{\alpha}_1$, $n \geq j > j' \geq 0$, the edge between the vertices $\bar{p}\bar{X}\mu$ and $\bar{p}\bar{X}\mu\nu$ has the following tuple as its *colour*:

$$\left(p_{i,j} X_{i,j}, p_{i,j'} X_{i,j'}, \text{DS}(\bar{p}\bar{X}\mu), \text{DS}(\bar{p}\bar{X}\mu\nu), (\text{DS}(q\nu), \text{ES}(q\nu))_{q \in \text{DS}(\bar{p}\bar{X}\mu)}, \mathbb{Q}_{\neq}, \mathbb{Q}_0 \right)$$

where $\mathbb{Q}_{\neq} = \{q' \in \text{DS}(\bar{p}\bar{X}\mu) \mid q'\nu\rho \not\sim q'\rho\}$ and $\mathbb{Q}_0 = \{q' \in \mathbb{Q}_{\neq} \mid q'\nu\rho \sim_0 q'\rho\}$ (and $p_{i,j} X_{i,j}, p_{i,j'} X_{i,j'}$ are taken from (5)).

Since the set of colours is bounded (by a constant determined by \mathcal{M}), Ramsey's theorem yields a bound B guaranteeing that there is a monochromatic clique of size 3 among the vertices $\bar{p}\bar{X}, \bar{p}\bar{X}\bar{\alpha}_n, \bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1}, \dots, \bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1} \cdots \bar{\alpha}_{n-B}$. (We have soundly chosen i so that $n = n_i$ is bigger than B .) We fix such a monochromatic clique MC, denoting its 3 vertices as

$$\bar{p}\bar{X}\beta, \bar{p}\bar{X}\beta\gamma, \bar{p}\bar{X}\beta\gamma\bar{\gamma}, \text{ and its colour as } C = (p'X', p'X', \bar{Q}, \bar{Q}, (\mathcal{D}_q, \mathcal{E}_q)_{q \in \bar{Q}}, Q', Q_0).$$

This is sound, since the fact that both edges $\{\bar{p}\bar{X}\beta, \bar{p}\bar{X}\beta\gamma\}$ and $\{\bar{p}\bar{X}\beta\gamma, \bar{p}\bar{X}\beta\gamma\bar{\gamma}\}$ have the same colour entails that the first component in this colour is the same as the second component, and the third component is the same as the fourth component.

We now show that the conditions 1–5 are satisfied for the presentation of $\bar{p}\bar{X}\bar{\alpha}_n\bar{\alpha}_{n-1} \cdots \bar{\alpha}_1$ as $\bar{p}\bar{X}\beta\gamma\delta$, where $\delta = \bar{\gamma}\bar{\alpha}_k\bar{\alpha}_{k-1} \cdots \bar{\alpha}_1$ for the respective k .

Conditions 1 and 2 are trivial (due to the colour C).

Condition 3: Let $q \in \bar{Q}$ and $\text{ES}(q\gamma) = \{q'\}$ (hence also $q' \in \bar{Q}$). Then $\mathcal{E}_q = \text{ES}(q\gamma) = \text{ES}(q\gamma\bar{\gamma}) = \{q'\}$ (since MC is monochromatic). This entails $\text{ES}(q'\bar{\gamma}) = \{q'\}$, hence $\mathcal{E}_{q'} = \{q'\}$, which in turn entails $\text{ES}(q'\gamma) = \{q'\}$.

Condition 4: We first note a general fact: $\text{DS}(p\mu\nu) = \bigcup_{q \in \text{DS}(p\mu)} \text{DS}(q\nu)$. Since $\bar{Q} = \text{DS}(\bar{p}\bar{X}\beta) = \text{DS}(\bar{p}\bar{X}\beta\gamma) = \text{DS}(\bar{p}\bar{X}\beta\gamma\bar{\gamma})$, for each $q' \in \bar{Q}$ there is thus $q \in \bar{Q}$ such that $q' \in \mathcal{D}_q$. We also have the following “transitivity”: if $q_1, q_2, q_3 \in \bar{Q}$, $q_1 \in \mathcal{D}_{q_2}$, and $q_2 \in \mathcal{D}_{q_3}$,

63:12 The Simplest Non-Regular Deterministic Context-Free Language

then $q_1 \in \mathcal{D}_{q_3}$ (since MC is monochromatic). For any $q' \in \bar{Q}$ there is clearly a “chain” $q' = q_1, q_2, q_3, \dots, q_\ell$ where $\ell > 1$, $q_j \in \mathcal{D}_{q_{j+1}}$ for all $j \in [1, \ell-1]$, and $q_j = q_\ell$ for some $j < \ell$. By the above transitivity, q_ℓ is self-containing ($q_\ell \in \mathcal{D}_{q_\ell}$ and thus $q_\ell \in \text{DS}(q_\ell\gamma)$) and $q' \in \mathcal{D}_{q_\ell}$ (hence $q' \in \text{DS}(q_\ell\gamma)$).

Condition 5: For any three configurations at least two belong to the same \sim_0 -class. Since the edges among the vertices $\bar{p}\bar{X}\beta, \bar{p}\bar{X}\beta\gamma, \bar{p}\bar{X}\beta\gamma\bar{\gamma}$ have the same Q'_0 in their colour C, we get that $Q'_0 = Q'$, and thus also $q'\gamma\delta \sim_0 q'\delta$ for all $q' \in \bar{Q}$ such that $q'\gamma\delta \not\sim q'\delta$. Now if for all $q' \in \bar{Q}$ we had $q'\gamma\delta \sim q'\delta$ (which includes the case $\bar{Q} = \emptyset$), then we would get a contradiction with our choice of (5) since it could have been shortened as in Claim 10. \triangleleft

Now we state a weaker version of Lemma 5:

\triangleright **Claim 13.** There are $v \in \Sigma^*, x, w, y, z \in \Sigma^+, p, q \in Q, X \in \Gamma, \gamma \in \Gamma^+, \delta \in \Gamma^*$ such that $p_0X_0 \xrightarrow{v} pX\delta, pX \xrightarrow{x} pX\gamma, pX \xrightarrow{w} q, q\gamma \xrightarrow{y} q$, and

- either $z \in \mathcal{L}(q\delta)$ and $z \notin \mathcal{L}(q\gamma^\ell\delta)$ for all $\ell > 0$,
- or $z \notin \mathcal{L}(q\delta)$ and $z \in \mathcal{L}(q\gamma^\ell\delta)$ for all $\ell > 0$.

Proof. We fix one $\bar{p}\bar{X}\beta\gamma\delta$ guaranteed by Claim 12 (satisfying the respective conditions 1–5). There are $v \in \Sigma^*, x, w, y, \bar{z} \in \Sigma^+, p, q \in Q, X \in \Gamma, \gamma \in \Gamma^+, \delta \in \Gamma^*, q' \in \text{DS}(q\gamma)$ such that

$p_0X_0 \xrightarrow{v} pX\delta, pX \xrightarrow{x} pX\gamma, pX \xrightarrow{w} q, q\gamma \xrightarrow{y} q$, and $\mathcal{L}(q'\gamma\delta)$ and $\mathcal{L}(q'\delta)$ differ on \bar{z} (i.e., $\bar{z} \in (\mathcal{L}(q'\gamma\delta) \setminus \mathcal{L}(q'\delta)) \cup (\mathcal{L}(q'\delta) \setminus \mathcal{L}(q'\gamma\delta))$).

Indeed: The respective computation (5) can be written $p_0X_0 \xrightarrow{v} pX\delta \xrightarrow{x} pX\gamma\delta \xrightarrow{w'} \bar{p}\bar{X}\beta\gamma\delta$ where x and γ are nonempty. The claimed q' and [nonempty] \bar{z} are guaranteed by 5 in Claim 12, and q is a respective self-containing state from 4. Since $q \in \text{DS}(\bar{p}\bar{X}\beta)$ and $q \in \text{DS}(q\gamma)$, we get $pX\gamma\delta \xrightarrow{w''} q\gamma\delta \xrightarrow{y} q\delta$, where $w'' \neq \varepsilon$. We also have $y \neq \varepsilon$, since otherwise $\text{DS}(q\gamma) = \text{ES}(q\gamma) = \{q\}$, $q' = q$, and we could not have $q\gamma\delta \not\sim q\delta$ and $q\gamma\delta \sim_0 q\delta$.

Since $q' \in \text{DS}(q\gamma)$, we can fix z' such that $q\gamma \xrightarrow{z'} q'$. Hence the languages $\mathcal{L}(q\gamma\gamma\delta)$ and $\mathcal{L}(q\gamma\delta)$ differ on $z = z'\bar{z}$; more generally, $\mathcal{L}(q\gamma^{\ell+1}\gamma\delta)$ and $\mathcal{L}(q\gamma^\ell\gamma\delta)$ differ on $y^\ell z$ for all $\ell \geq 0$. Now we aim to find out for which ℓ we have $z \in \mathcal{L}(q\gamma^\ell\delta)$.

We recall that $\bar{Q} = \text{DS}(\bar{p}\bar{X}\beta) = \text{DS}(\bar{p}\bar{X}\beta\gamma)$; hence $\bigcup_{\bar{q} \in \bar{Q}} \text{DS}(\bar{q}\gamma) = \bar{Q}$. Since $q \in \bar{Q}$, we get that $\text{DS}(q\gamma^d) \subseteq \bar{Q}$ for all $d \in \mathbb{N}$ (by induction). We now distinguish two cases:

1. For each prefix z_1 of z and each $d \leq |z|$ we have: if $q\gamma^d \xrightarrow{z_1} \bar{q}$, then $\text{ES}(\bar{q}\gamma) = \emptyset$.
2. There are a prefix z_1 of z , $d \leq |z|$, and $\bar{q}, q'' \in \bar{Q}$ such that $q\gamma^d \xrightarrow{z_1} \bar{q}$ and $\text{ES}(\bar{q}\gamma) = \{q''\}$.

In the case 1 we clearly have either $\forall \ell > |z| : z \in \mathcal{L}(q\gamma^\ell\delta)$ or $\forall \ell > |z| : z \notin \mathcal{L}(q\gamma^\ell\delta)$ (here δ plays no role). In the case 2 we recall that $\bar{q}\gamma \xrightarrow{\varepsilon} q''$ entails that $\bar{q}\gamma^k\delta \xrightarrow{\varepsilon} q''\delta$ for all $k \geq 1$ (since $\text{ES}(q''\gamma) = \{q''\}$ by 3 in Claim 12). Hence we have either $\forall \ell > |z| + 1 : z \in \mathcal{L}(q\gamma^\ell\delta)$ or $\forall \ell > |z| + 1 : z \notin \mathcal{L}(q\gamma^\ell\delta)$.

Since $\mathcal{L}(q\gamma^2\delta)$ and $\mathcal{L}(q\gamma\delta)$ differ on z , we deduce that there is $\ell_0 \geq 1$ such that either $z \in \mathcal{L}(q\gamma^{\ell_0}\delta)$ and $z \notin \mathcal{L}(q\gamma^\ell\delta)$ for all $\ell > \ell_0$, or $z \notin \mathcal{L}(q\gamma^{\ell_0}\delta)$ and $z \in \mathcal{L}(q\gamma^\ell\delta)$ for all $\ell > \ell_0$. Hence for $\bar{\delta} = \gamma^{\ell_0}\delta$ we have either $z \in \mathcal{L}(q\bar{\delta})$ and $z \notin \mathcal{L}(q\gamma^\ell\bar{\delta})$ for all $\ell > 0$, or $z \notin \mathcal{L}(q\bar{\delta})$ and $z \in \mathcal{L}(q\gamma^\ell\bar{\delta})$ for all $\ell > 0$. Since for $\bar{v} = vx^{\ell_0}$ we have $p_0X_0 \xrightarrow{\bar{v}} pX\bar{\delta}$, the claim is proven. \triangleleft

Claim 13 shows that there is $L' \in \{L, \bar{L}\}$ such that $vx^mwy^mz \in L'$ and $vx^mwy^n z \notin L'$ for $m > n$, which is a weaker version of Lemma 5. To handle the case $m < n$, we have to find out for which ℓ we have $y^\ell z \in \mathcal{L}(q\delta)$. We thus look at the computation from $q\delta$ on the infinite word y^ω (recalling our convention that this computation is infinite, stepwise reading the word $yyy\cdots$), and use the obvious fact that after a prefix this computation becomes “periodic” (either cycling among finitely many configurations, or increasing the stack forever).

▷ Claim 14. For any configuration $q\delta$ and words y, z there are numbers $s \geq 0$ (“shift”) and $P > 0$ (“period”) such that for all $\ell \geq s$ the remainder $(\ell \bmod P)$ determines whether or not $\mathcal{L}(q\delta) \ni y^\ell z$.

Proof. We assume $y \neq \varepsilon$ (otherwise the claim is trivial). For the infinite computation from $q\delta$ on $yyy \dots$ there are obviously $k_1 \geq 0$, $k_2 > 0$, $\bar{q} \in Q$, and $\rho, \mu, \nu \in \Gamma^*$ such that the computation can be written $q\delta \xrightarrow{y^{k_1}} \bar{q}\rho\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\mu\nu \xrightarrow{y^{k_2}} \bar{q}\rho\mu\mu\mu\nu \xrightarrow{y^{k_2}} \dots$ where $\bar{q}\rho \xrightarrow{y^{k_2}} \bar{q}\rho\mu$. (We have $\mu = \varepsilon$ if the computation visits only finitely many configurations, and otherwise we consider the stair-factorization of the computation.)

For each $j \in [0, k_2-1]$ we put $\bar{q}\rho \xrightarrow{y^j} \bar{q}_j\rho_j$, and we have two possible cases:

1. There is $d_0 \geq 0$ such that for all $d \geq d_0$ performing z from $\bar{q}_j\rho_j\mu^d\nu$ does not reach ν at the bottom.
2. There are $d_0 \geq 0$, a prefix z' of z , $q' \in Q$, and $\bar{d} \in [1, |Q|]$ such that $\bar{q}_j\rho_j\mu^{d_0} \xrightarrow{z'} q'$ and $q'\mu^{\bar{d}} \xrightarrow{\varepsilon} q'$.

In the case 1 either $\mathcal{L}(q\delta) \ni y^{d \cdot k_2 + j} z$ for all $d \geq d_0$, or $\mathcal{L}(q\delta) \not\ni y^{d \cdot k_2 + j} z$ for all $d \geq d_0$.

In the case 2, for each $d \geq 0$ we have $q'\mu^d \xrightarrow{\varepsilon} q_d$ where $q_{d_1} = q_{d_2}$ if $d_1 \equiv d_2 \pmod{\bar{d}}$. Hence for each $d \geq d_0$, the (non)membership of $y^{d \cdot k_2 + j} z$ in $\mathcal{L}(q\delta)$ is determined by $(d \bmod \bar{d})$.

The claim is thus clear. \triangleleft

Now we finish the proof of Lemma 5. We take the notation from Claim 13; for the respective $q\delta, y, z$ we add s, P from Claim 14. Let k_0 be a multiple of P that is bigger than s . We now view $x^{k_0}, y^{k_0}, \gamma^{k_0}$ as new x, y, γ , respectively. Claims 13 and 14 now yield the statement of Lemma 5.

4 Conclusion and Open Problems

In this paper, we have introduced a new notion of the \mathcal{C} -simple problem that reduces to each problem in \mathcal{C} , being thus a conceptual counterpart to the \mathcal{C} -hard problem to which each problem in \mathcal{C} reduces. We have illustrated this concept on the definition of the DCFL'-simple problem that reduces to each DCFL' language under the truth-table reduction by Mealy machines. We have proven that the DCFL' language $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ is DCFL'-simple, and thus represents the simplest languages in the class DCFL'. This result finds its application in expanding the known lower bound for $L_{\#}$, namely that $L_{\#}$ cannot be recognized by the neural network model 1ANN, to all DCFL' languages. Moreover, the class DCFLS of DCFL'-simple problems containing the regular languages is a strict subclass of DCFL and has similar closure properties as DCFL.

We note that the hardest context-free language L_0 by Greibach [3], where each L in CFL is an inverse homomorphic image of L_0 or $L_0 \setminus \{\varepsilon\}$, can be viewed as CFL-hard w.r.t. a many-one reduction based on Mealy machines realizing the respective homomorphisms. Our aims in the definition of DCFL'-simple problems cannot be achieved by such a many-one reduction, hence we have generalized it to a truth-table reduction. We can alternatively consider a general Turing reduction that is implemented by a Mealy machine which queries the oracle at special query states, each associated with a corresponding query suffix, while its next transition from the query state depends on the given oracle answer. The oracle Mealy machine then accepts an input word if it reaches an accept state after reading the input. The language $L_{\#}$ proves to be DCFL'-simple under this Turing reduction allowing for an unbounded number of online oracle queries; this can be shown by Claim 13 (a weaker version of Lemma 5).

It is natural to try extending our result to non-regular nondeterministic (or at least unambiguous) context-free languages, by possibly showing that $L_{\#}$ is CFL'-simple. Another important challenge for further research is looking for \mathcal{C} -simple problems for other complexity classes \mathcal{C} and suitable reductions. This could provide an effective tool for strengthening lower-bounds results known for single problems to the whole classes of problems, which deserves a deeper study.

References

- 1 M. Anabtawi, S. Hassan, Christos A. Kapoutsis, and M. Zakzok. An oracle hierarchy for small one-way finite automata. In *Proceedings of LATA 2019*, LNCS 11417, pages 57–69. Springer, 2019. doi:10.1007/978-3-030-13435-8_4.
- 2 Jean Berstel and Luc Boasson. Context-free languages. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 59–102. Elsevier and MIT Press, 1990. doi:10.1016/b978-0-444-88074-1.50007-x.
- 3 Sheila A. Greibach. The hardest context-free language. *SIAM J. Comput.*, 2(4):304–310, 1973. doi:10.1137/0202025.
- 4 John E. Hopcroft and Jeffrey D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley, 1969. URL: <https://www.worldcat.org/oclc/00005012>.
- 5 Petr Jančar. Deciding semantic finiteness of pushdown processes and first-order grammars w.r.t. bisimulation equivalence. *J. Comput. Syst. Sci.*, 109:22–44, 2020. doi:10.1016/j.jcss.2019.10.002.
- 6 Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel. On monotonic automata with a restart operation. *J. Autom. Lang. Comb.*, 4(4):287–311, 1999. doi:10.25596/jalc-1999-287.
- 7 František Mráz, Dana Pardubská, Martin Plátek, and Jiří Šíma. Pumping deterministic monotone restarting automata and DCFL. In *Proceedings of ITAT 2020*, CEUR Workshop Proceedings 2718, pages 51–58, 2020. URL: <http://ceur-ws.org/Vol-2718/paper13.pdf>.
- 8 Klaus Reinhardt. Hierarchies over the context-free languages. In *Proceedings of IMYCS 1990*, LNCS 464, pages 214–224. Springer, 1990. doi:10.1007/3-540-53414-8_44.
- 9 Hava T. Siegelmann. *Neural networks and analog computation – Beyond the Turing limit*. Birkhäuser, 1999.
- 10 Jiří Šíma. Analog neuron hierarchy. *Neural Netw.*, 128:199–215, 2020. doi:10.1016/j.neunet.2020.05.006.
- 11 Jiří Šíma. Stronger separation of analog neuron hierarchy by deterministic context-free languages, 2021. (submitted to a journal). arXiv:2102.01633.
- 12 Jiří Šíma and Pekka Orponen. General-purpose computation with neural networks: A survey of complexity theoretic results. *Neural Comput.*, 15(12):2727–2778, 2003. doi:10.1162/089976603322518731.
- 13 Jiří Šíma and Martin Plátek. One analog neuron cannot recognize deterministic context-free languages. In *Proceedings of ICONIP 2019, Part III*, LNCS 11955, pages 77–89. Springer, 2019. doi:10.1007/978-3-030-36718-3_7.
- 14 Tomoyuki Yamakami. Oracle pushdown automata, nondeterministic reducibilities, and the hierarchy over the family of context-free languages. In *Proceedings of SOFSEM 2014*, LNCS 8327, pages 514–525. Springer, 2014. (full version arXiv:1303.1717). doi:10.1007/978-3-319-04298-5_45.

A Informal overview of Lemma 5 and of its proof

Given a DPDA \mathcal{M} accepting a non-regular language $L = \mathcal{L}(\mathcal{M}) = \mathcal{L}(p_0 X_0) \subseteq \Sigma^*$, Lemma 5 claims that there is a word $v \in \Sigma^*$ and nonempty words $x, w, y, z \in \Sigma^+$ with the properties depicted in Figure 1, which entail the following conditions:

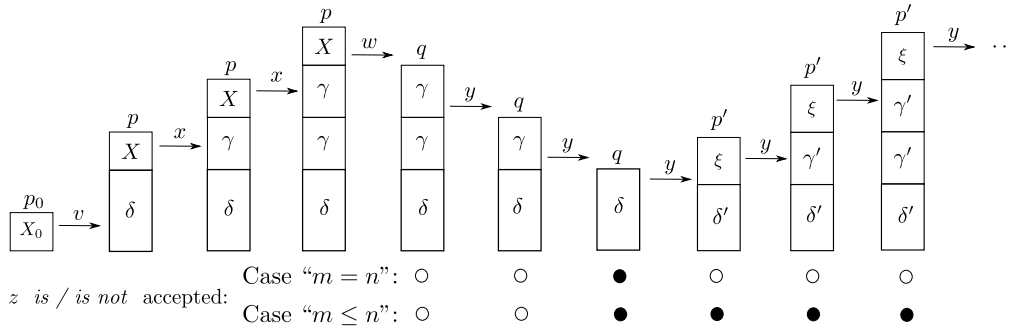


Figure 1 DPDA computation scheme where either $vx^mwy^n z$ is (or is not) accepted iff $m = n$, or $vx^mwy^n z$ is (or is not) accepted iff $m \leq n$.

1. (the pumping condition)

$p_0X_0 \xrightarrow{v} pX\delta \xrightarrow{x^m} pX\gamma^m\delta \xrightarrow{w} q\gamma^m\delta \xrightarrow{y^m} q\delta$ for all $m \geq 0$ (since $pX \xrightarrow{x} pX\gamma$ and $q\gamma \xrightarrow{y} q$); hence $z \in \mathcal{L}(q\delta)$ entails $vx^mwy^m z \in L$ for all $m \geq 0$, and $z \notin \mathcal{L}(q\delta)$ entails $vx^mwy^m z \notin L$ for all $m \geq 0$;

2. (the prefix condition)

the prefix differs from $q\delta$ on z in the sense that the languages of all configurations reachable by vx^mwy^n where $m > n$ differ from $\mathcal{L}(q\delta)$ on z ; referring to Figure 1, $z \in \mathcal{L}(q\gamma^k\delta) \Delta \mathcal{L}(q\delta)$ for all $k > 0$, where $A \Delta B$ denotes $(A \setminus B) \cup (B \setminus A)$;

3. (the suffix condition)

the suffix (all configurations reachable by vx^mwy^n where $m < n$) either differs from, or coincides with, $q\delta$ on z ; referring to Figure 1, either $z \in \mathcal{L}(q\delta) \Delta \mathcal{L}(p'\xi(\gamma')^k\delta')$ for all $k \geq 0$, or $z \in \mathcal{L}(q\delta) \cap \mathcal{L}(p'\xi(\gamma')^k\delta')$ for all $k \geq 0$.

The prefix condition 2 implies that the stack segment γ is nonempty (while γ' might be empty). The conditions also imply that $q \in DS(q\gamma)$ and $ES(q\gamma) = \emptyset$, when we use the following definitions: $DS(r\alpha) = \{r' \mid r\alpha \xrightarrow{u} r' \text{ for some } u \in \Sigma^*\}$ (the “down-states” of $r\alpha$) and $ES(r\alpha) = \{r' \mid r\alpha \xrightarrow{\varepsilon} r' \}$ (which is either the empty set or a singleton containing the down-state reached by a sequence of ε -poppings from $r\alpha$). Hence $ES(r\alpha) \subseteq DS(r\alpha)$, and $ES(r\alpha) \neq \emptyset$ entails $ES(r\alpha) = DS(r\alpha) = \{r'\}$ for some r' . Figure 2 depicts an example of $DS(pX\gamma^5)$, using the obvious compositional approach based on $DS(pX)$ and $DS(q_i\gamma)$ and $ES(q_i\gamma)$ where $i \in \{1, 2, 3, 4, 5\}$, assuming that the state set of \mathcal{M} is $Q = \{q_1, q_2, q_3, q_4, q_5\}$ and $p = q_2$. (Here the stack is presented horizontally.)

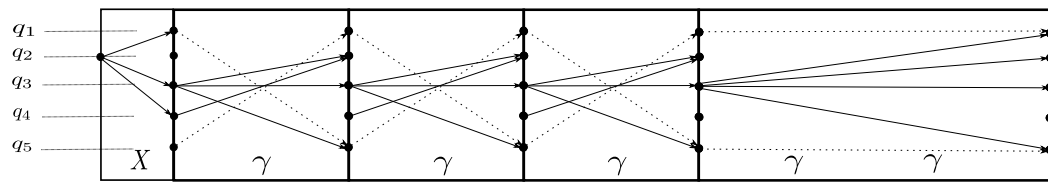
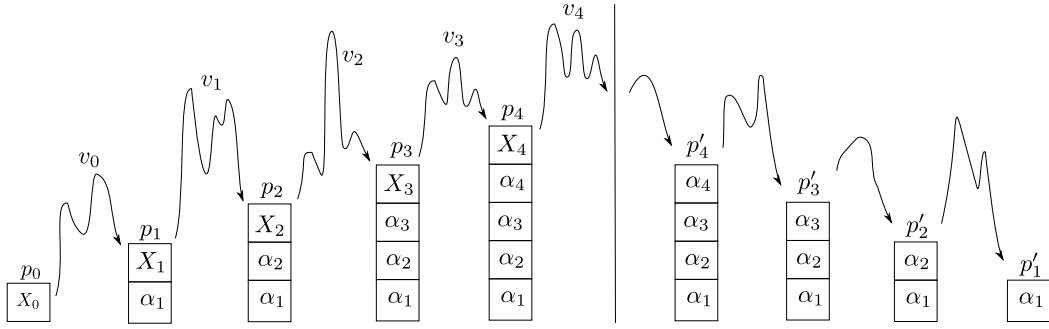


Figure 2 Each directed path from the leftmost black point to the rightmost upper point shows that $q_1 \in DS(q_2X\gamma^5)$. The completely-dashed paths correspond to ε -sequences; e.g. $ES(q_5\gamma\gamma) = \{q_5\}$.

Getting tuples (v, x, w, y, z) satisfying the pumping condition 1

Since $L = \mathcal{L}(p_0X_0)$ is not regular, it is clear that from p_0X_0 the computations of \mathcal{M} can reach configurations with arbitrary stack-heights, more precisely configurations with arbitrarily long erasable stack-tops. (The stack-top α in a configuration $p\alpha\beta$ is erasable if $DS(p\alpha) \neq \emptyset$.)



■ **Figure 3** Stair-factorization.

Moreover, such long stack-tops must be also erasable by using many “solid-line” segments that use visible (i.e. non- ε) steps. Indeed: if all possible stack-erasings would in principle go along the dashed lines, i.e. by ε -popping, like from $q_1\gamma^k$ in Figure 2, then this would also entail regularity of L , since even long (erasable) stacks could be replaced with their equivalents of bounded lengths in such cases.

Using the above observations, it is standard to derive the existence of various tuples (v, x, w, y, z) satisfying the pumping condition 1. A crucial fact is that any computation $p_0X_0 \xrightarrow{u} r\alpha$ where the stack-content α is long can be *stair-factorized* into a long sequence of “stairs”, as depicted on the left in Figure 3: here $p_iX_i \xrightarrow{v_i} p_{i+1}X_{i+1}\alpha_{i+1}$ and α_{i+1} is a nonempty suffix of the right-hand side of a rewriting rule of \mathcal{M} (for $i = 0, 1, 2, \dots$). If a (long) stack $\alpha_k\alpha_{k-1}\dots\alpha_1$ is first built and then its (long) top $\alpha_k\alpha_{k-1}\dots\alpha_{j+1}$ gets erased, we let p'_i denote the state in which $\alpha_i\alpha_{i-1}\dots\alpha_j\alpha_{j-1}\dots\alpha_1$ is exposed during this erasing (for $i = k, k-1, \dots, j$); such p'_i are depicted on the right in Figure 3, assuming $j = 1$. By the pigeonhole principle, a triple (p, X, p') repeats in a sufficiently long sequence $(p_j, X_j, p'_j), (p_{j+1}, X_{j+1}, p'_{j+1}), \dots, (p_k, X_k, p'_k)$, which naturally yields a “pumping tuple” (v, x, w, y, z) .

Pumping-operation on (v, x, w, y, z) (preserving the conditions 1, 2, 3 that hold)

Looking at Figure 1, we observe that if the pumping condition 1 holds for a tuple (v, x, w, y, z) , then it is preserved by the *pumping-operation* on (v, x, w, y, z) that consists in replacing x and y with their “multiples” x^{k_0} and y^{k_0} , for any $k_0 \geq 1$. Moreover, if (v, x, w, y, z) also happens to satisfy the prefix condition 2, then also this condition is preserved by the pumping-operation (for any $k_0 \geq 1$). The same is true for the suffix condition 3.

Establishing the suffix condition 3 (by the pumping-operation for suitable k_0)

Given (v, x, w, y, z) that satisfies the pumping condition 1, we now show that the pumping-operation (for an appropriate number k_0) establishes the suffix condition 3. First we observe that if \mathcal{M} starts in $q\delta$ and processes $y^\omega = yyy\dots$, then the respective infinite computation necessarily enters a “cycle” after a “prelude”. This is depicted in Figure 1, but there both the prelude and the cycle process the word y . Generally, we would get a prelude $q\delta \xrightarrow{y^{k_1}} p'\xi\delta'$ and a cycle $p'\xi\delta' \xrightarrow{y^{k_2}} p'\xi\gamma'\delta'$ (where $p'\xi \xrightarrow{y^{k_2}} p'\xi\gamma'$ and γ' might be empty) for some numbers k_1, k_2 (where $k_2 > 0$). We now show that the set

$$A = \{\ell \in \mathbb{N} \mid z \in \mathcal{L}(C_\ell)\} \text{ where } C_\ell \text{ are the configurations satisfying } q\delta \xrightarrow{y^\ell} C_\ell \quad (6)$$

is ultimately periodic; i.e., for a *shift* $s \geq 0$ and a *period* $P > 0$ we have that for all $\ell \geq s$ the remainder $(\ell \bmod P)$ determines whether or not $\ell \in A$. Generally we cannot simply take $P = k_2$ as a suitable period, since z might “embark” on popping the γ' -segments along “dashed paths”: processing a prefix z_1 of z from $p'\xi\gamma'\gamma'\cdots\gamma'\delta'$ might reach a configuration $q'\gamma'\gamma'\cdots\gamma'\delta'$ like $q_1\gamma\gamma\gamma\cdots$ in Figure 2, in which case we have $q'\gamma'\gamma'\cdots\gamma'\delta' \xrightarrow{\varepsilon} r\delta'$, and it is the state r in which the bottom δ' is reached that determines whether z is accepted or not (i.e., whether $z_2 \in \mathcal{L}(r\delta')$ when $z = z_1z_2$). We thus might need to choose P as a multiple of k_2 , guaranteeing that the above mentioned state r (in which δ' is reached) is also repeating with the period P .

Having a shift s and a period P characterizing the ultimate periodicity of the set A defined by (6), we choose $k_0 \geq s$ that is a multiple of P . Then replacing x and y with x^{k_0} and y^{k_0} indeed guarantees the suffix condition 3; an important point is that the “suffix” might *differ from, or coincide with, $q\delta$* on z .

Establishing the prefix condition 2

Given a tuple (v, x, w, y, z) satisfying the pumping condition 1, if we aim to establish the prefix condition 2 by the pumping-operation, then it is natural to consider the “prefix-counterpart” of (6), namely the set

$$A' = \{\ell \in \mathbb{N} \mid z \in \mathcal{L}(q\gamma^\ell\delta)\} \quad (7)$$

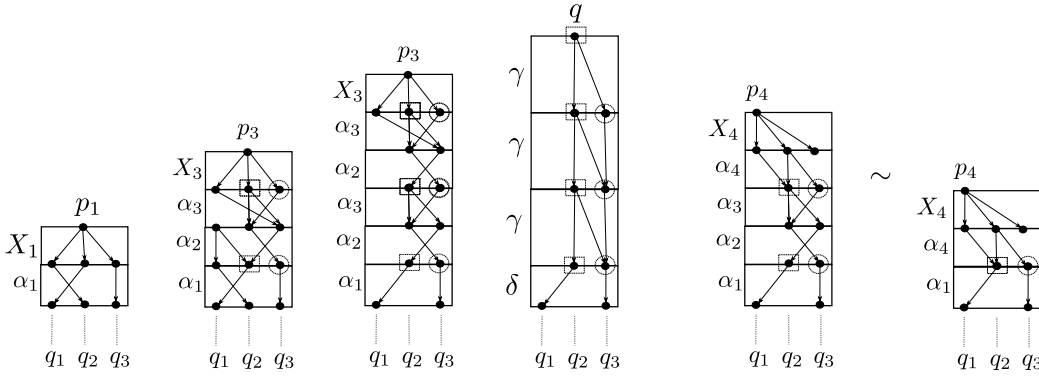
(recall Figure 1). It is again clear that A' is ultimately periodic, but a problem is that we have to guarantee that the “prefix” has to *differ from $q\delta$* on z (unlike the “suffix” that can also coincide).

We now show that if A' is nontrivial ($\emptyset \subsetneq A' \subsetneq \mathbb{N}$), then we can establish the prefix condition 2 easily. Let s be the shift and P the period of a presentation of A' as an ultimately periodic set, and let $i_0 \in A'$ and $i_1 \notin A'$. Let $k_0 \geq \max\{i_0, i_1, s\}$ be a multiple of P , and let $j \in \{0, 1\}$ be such that i_j differs from k_0 on the membership in A' . Instead of (v, x, w, y, z) we now take (v', x', w, y', z) where $v' = vx^{i_j}$, $x' = x^{k_0}$, $y' = y^{k_0}$. Referring to Figure 1, by this change δ is replaced with $\bar{\delta} = \gamma^{i_j}\delta$, and γ is replaced with $\bar{\gamma} = \gamma^{k_0}$. We have $z \in \mathcal{L}(q\bar{\delta}) \triangle \mathcal{L}(q(\bar{\gamma})^k\bar{\delta})$ for all $k > 0$; hence the prefix condition 2 is indeed established.

It remains to explore if we can have the case that for each tuple (v, x, w, y, z) satisfying the pumping condition 1 the “prefix” set A' defined by (7) (when referring to the notation of Figure 1) is trivial. Since we can choose z freely, this case would, in fact, entail that $q\delta \sim q\gamma\delta \sim q\gamma\gamma\delta \sim \cdots$, where $C \sim C'$ stands for $\mathcal{L}(C) = \mathcal{L}(C')$ for any configurations C, C' . We now show that this case cannot happen since the language $L = \mathcal{L}(q_0X_0)$ is non-regular.

First, it is straightforward to derive that we can fix a *crucial infinite computation of \mathcal{M} from p_0X_0* , processing some word $a_1a_2a_3\cdots$, whose stair-factorization has infinitely many stairs and each stair represents its own equivalence class of \sim . We can view the left part of Figure 3 as a prefix of this crucial computation; we thus have $p_iX_i\alpha_i\alpha_{i-1}\cdots\alpha_1 \not\sim p_jX_j\alpha_j\alpha_{j-1}\cdots\alpha_1$ for all $i \neq j$. (The existence of such an infinite computation follows by the fact that the set of left quotients $\{u \setminus L \mid u \in \Sigma^*\}$, where $u \setminus L = \{u' \in \Sigma^* \mid uu' \in L\}$, is infinite since L is non-regular, and by König’s lemma since the tree of all computations of \mathcal{M} from p_0X_0 is finitely branching.)

We are not done, since even in this crucial infinite computation (with pairwise non-equivalent stairs) a “pumping” tuple (v, x, w, y, z) derived by the above-mentioned pigeonhole principle might not guarantee that $q\delta \not\sim q\gamma\delta$ (and we might have $q\delta \sim q\gamma\delta \sim q\gamma\gamma\delta \sim \cdots$). For instance, let us assume that in Figure 3 we have $p_1X_1 = p_3X_3$, and that $Q = \{q_1, q_2, q_3\}$ as depicted in Figure 4. We can have $q_2\alpha_1 \sim q_2\alpha_3\alpha_2\alpha_1$ (as denoted by the rectangles in



■ **Figure 4** Shortening of configurations (here $p_1X_1 = p_3X_3$).

Figure 4) and $q_3\alpha_1 \sim q_3\alpha_3\alpha_2\alpha_1$ (as denoted by the circles), but $q_1\alpha_1 \not\sim q_1\alpha_3\alpha_2\alpha_1$ (which causes that $p_1X_1\alpha_1 \not\sim p_3X_3\alpha_3\alpha_2\alpha_1$). By putting $pX = p_1X_1 = p_3X_3$, $\delta = \alpha_1$, $x = v_1v_2$ (referring to Figure 3), and $\gamma = \alpha_3\alpha_2$, we get a “pumping” $p_0X_0 \xrightarrow{v_0} pX\delta \xrightarrow{x} pX\gamma\delta \xrightarrow{x} pX\gamma\gamma\delta$ where $pX\gamma\gamma\delta$ is depicted as the third configuration in Figure 4. (We have omitted unreachable “black points.”) Here we indeed have $q\delta \sim q\gamma\delta \sim q\gamma\gamma\delta \sim \dots$, as is highlighted by the fourth configuration in Figure 4.

In our example we can also note that some configurations in the crucial infinite computation might be safely shortened while their equivalence classes are preserved. This is depicted on the right in Figure 4: we have $p_0X_0 \xrightarrow{v_0v_1v_2v_3} p_4X_4\alpha_4\alpha_3\alpha_2\alpha_1$, but we obviously have $p_4X_4\alpha_4\alpha_3\alpha_2\alpha_1 \sim p_4X_4\alpha_4\alpha_1$; this shorter representant of the equivalence class of $p_4X_4\alpha_4\alpha_3\alpha_2\alpha_1$ is reachable by omitting v_1v_2 , i.e., $p_0X_0 \xrightarrow{v_0v_3} p_4X_4\alpha_4\alpha_1$.

Nevertheless, the crucial computation visits infinitely many equivalence classes, so the sizes of the stair-configurations $p_iX_i\alpha_i\alpha_{i-1}\dots\alpha_1$ must grow above any bound even when we first shorten them maximally in the (repeated) described way. Let us now fix a stair-configuration that has been maximally shortened in the above way and is still sufficiently long. By straightforward combinatorial arguments (that can be presented as an application of Ramsey’s theorem to avoid tedious technicalities) we can derive that this (shortened) configuration can be written as $\bar{p}\bar{X}\bar{\beta}\gamma\delta$ where γ is nonempty, and

- γ can be pumped (having the same “lower” p_jX_j and “upper” $p_{j'}X_{j'}$, like $\gamma = \alpha_3\alpha_2$ in Figure 4);
- the sets $\text{DS}(\bar{p}\bar{X}\bar{\beta})$ and $\text{DS}(\bar{p}\bar{X}\bar{\beta}\gamma)$ are equal, further being denoted by \bar{Q} (e.g., in Figure 4 $p_4X_4\alpha_4\alpha_3\alpha_2\alpha_1 = \bar{p}\bar{X}\bar{\beta}\gamma\delta$ where $\bar{Q} = \{q_2, q_3\}$);
- there is $q' \in \bar{Q}$ for which $\mathcal{L}(q'\gamma\delta)$ and $\mathcal{L}(q'\delta)$ differ on a nonempty word \bar{z} (e.g., now let the circled $q_3\gamma\delta$ and $q_3\delta$ in Figure 4 differ in this way, hence we can choose $q' = q_3$);
- moreover, this $q' \in \bar{Q}$ belongs to $\text{DS}(q\gamma)$ for some self-containing $q \in \bar{Q}$, where $q \in \bar{Q}$ is *self-containing* if $q \in \text{DS}(q\gamma)$ (let $q = q_2$ in our example, though here also q_3 is possible).

It is then clear that $q\gamma \xrightarrow{y} q$ and $q\gamma \xrightarrow{z'} q'$ for some nonempty y, z' ; this entails $q\gamma\gamma\delta \not\sim q\gamma\delta$ since they differ on $z = z'\bar{z}$. (This is sufficient for us even if we cannot deduce that $q\gamma\delta \not\sim q\delta$.)

A formal proof is given in the main part of the paper.




On the Hardness of Compressing Weights

Bart M. P. Jansen   

Eindhoven University of Technology, The Netherlands

Shivesh K. Roy   

Eindhoven University of Technology, The Netherlands

Michał Włodarczyk   

Eindhoven University of Technology, The Netherlands

Abstract

We investigate computational problems involving large weights through the lens of kernelization, which is a framework of polynomial-time preprocessing aimed at compressing the instance size. Our main focus is the weighted CLIQUE problem, where we are given an edge-weighted graph and the goal is to detect a clique of total weight equal to a prescribed value. We show that the weighted variant, parameterized by the number of vertices n , is significantly harder than the unweighted problem by presenting an $\mathcal{O}(n^{3-\varepsilon})$ lower bound on the size of the kernel, under the assumption that $\text{NP} \not\subseteq \text{coNP/poly}$. This lower bound is essentially tight: we show that we can reduce the problem to the case with weights bounded by $2^{\mathcal{O}(n)}$, which yields a randomized kernel of $\mathcal{O}(n^3)$ bits.

We generalize these results to the weighted d -UNIFORM HYPERCLIQUE problem, SUBSET SUM, and weighted variants of Boolean CONSTRAINT SATISFACTION PROBLEMS (CSPs). We also study weighted minimization problems and show that weight compression is easier when we only want to preserve the collection of optimal solutions. Namely, we show that for node-weighted VERTEX COVER on bipartite graphs it is possible to maintain the set of optimal solutions using integer weights from the range $[1, n]$, but if we want to maintain the ordering of the weights of all inclusion-minimal solutions, then weights as large as $2^{\Omega(n)}$ are necessary.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms; Theory of computation \rightarrow Problems, reductions and completeness

Keywords and phrases kernelization, compression, edge-weighted clique, constraint satisfaction problems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.64

Related Version *Full Version*: <https://arxiv.org/abs/2107.02554>

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803421, ReduceSearch).



1 Introduction

A prominent class of problems in algorithmic graph theory consist of finding a subgraph with certain properties in an input graph G , if one exists. Some variations of this problem can be solved in polynomial time (detecting a triangle), while the general problem is NP-complete since it generalizes the CLIQUE problem. In recent years, there has been an increasing interest in understanding the complexity of such subgraph detection problems in *weighted* graphs, where either the vertices or the edges are assigned integral weight values, and the goal is either to find a subgraph of a given form which optimizes the total weight of its elements, or alternatively, to find a subgraph whose total weight matches a prescribed value.

Incorporating weights in the problem definition can have a significant effect on computational complexity. For example, determining whether an unweighted n -vertex graph has a triangle can be done in time $\mathcal{O}(n^\omega)$ (where $\omega < 2.373$ is the exponent of matrix



© Bart M. P. Jansen, Shivesh K. Roy, and Michał Włodarczyk;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 64; pp. 64:1–64:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

multiplication) [14], while for the analogous weighted problem of finding a triangle of minimum edge-weight, no algorithm of running time $\mathcal{O}(n^{3-\varepsilon})$ is known for any $\varepsilon > 0$. Some popular conjectures in fine-grained complexity theory even postulate that no such algorithms exist [27]. Weights also have an effect on the best-possible exponential running times of algorithms solving NP-hard problems: the current-fastest algorithm for the NP-complete HAMILTONIAN CYCLE problem in undirected graphs runs in time $\mathcal{O}(1.66^n)$ [3], while for its weighted analogue, TRAVELING SALESPERSON, no algorithm with running time $\mathcal{O}((2 - \varepsilon)^n)$ is known for general undirected graphs (cf. [23]).

In this work we investigate how the presence of weights in a problem formulation affects the *compressibility* and *kernelization complexity* of NP-hard problems. Kernelization is a subfield of parameterized complexity [6, 9] that investigates how much a *polynomial-time* preprocessing algorithm can compress an instance of an NP-hard problem, without changing its answer, in terms of a chosen complexity parameter.

For a motivating example of kernelization, we consider the VERTEX COVER problem. For the unweighted variant, a kernelization algorithm based on the Nemhauser-Trotter theorem [25] can efficiently reduce an instance (G, k) of the decision problem, asking whether G has a vertex cover of size at most k , to an equivalent one (G', k') consisting of at most $2k$ vertices, which can therefore be encoded in $\mathcal{O}(k^2)$ bits via its adjacency matrix. In the language of parameterized complexity, the unweighted VERTEX COVER problem parameterized by the solution size k admits a kernelization (self-reduction) to an equivalent instance on $\mathcal{O}(k^2)$ bits. For the *weighted* variant of the problem, where an input additionally specifies a weight threshold $t \in \mathbb{N}_+$ and a weight function $w: V(G) \rightarrow \mathbb{N}_+$ on the vertices, and the question is whether there is a vertex cover of size at most k and weight at most t , the guarantee on the encoding size of the reduced instance is weaker. Etscheid et al. [10, Thm. 5] applied a powerful theorem of Frank and Tardős [12] to develop a polynomial-time algorithm to reduce any instance (G, w, k, t) of WEIGHTED VERTEX COVER to an equivalent one with $\mathcal{O}(k^2)$ edges, which nevertheless needs $\mathcal{O}(k^8)$ bits to encode due to potentially large numbers occurring as vertex weights. The WEIGHTED VERTEX COVER problem, parameterized by solution size k , therefore has a kernel of $\mathcal{O}(k^8)$ bits.

The overhead in the kernel size for the weighted problem is purely due to potentially large weights. This led Etscheid et al. [10] to ask in their conclusion whether this overhead in the kernelization sizes of weighted problems is necessary, or whether it can be avoided. As one of the main results of this paper, we will prove a lower bound showing that the kernelization complexity of some weighted problems is strictly larger than their unweighted counterparts.

Our results. We consider an edge-weighted variation of the CLIQUE problem, parameterized by the number of vertices n :

EXACT-EDGE-WEIGHT CLIQUE (EEWC)

Input: An undirected graph G , a weight function $w: E(G) \rightarrow \mathbb{N}_0$, and a target $t \in \mathbb{N}_0$.

Question: Does G have a clique of total edge-weight exactly t , i.e., a vertex set $S \subseteq V(G)$ such that $\{x, y\} \in E(G)$ for all distinct $x, y \in S$ and such that $\sum_{\{x, y\} \subseteq S} w(\{x, y\}) = t$?

Our formulation of EEWC does not constrain the cardinality of the clique. This formulation will be convenient for our purposes, but we remark that by adjusting the weight function it is possible to enforce that any solution clique S has a prescribed cardinality. Through such a cardinality restriction we can obtain a simple reduction from the problem with potentially negative weights to equivalent instances with weights from \mathbb{N}_0 , by increasing all weights by a suitably large value and adjusting t according to the prescribed cardinality.

Note that an instance of EEWC can be reduced to an equivalent one where G has all possible edges, by simply inserting each non-edge with a weight of $t + 1$. Hence the difficulty of the problem stems from achieving the given target weight t as the total weight of the edges spanned by S , not from the requirement that $G[S]$ must be a clique.

EEWC is a natural extension of ZERO-WEIGHT TRIANGLE [1], which has been studied because it inherits fine-grained hardness from both 3-SUM [29] and ALL PAIRS SHORTEST PATHS [28, Footnote 3]. EEWC has previously been considered by Abboud et al. [2] as an intermediate problem in their W[1]-membership reduction from k -SUM to k -CLIQUE. Vassilevska-Williams and Williams [29] considered a variation of this problem with weights drawn from a finite field. The related problem of detecting a triangle of negative edge weight is central in the field of fine-grained complexity for its subcubic equivalence [30] to ALL PAIRS SHORTEST PATHS. Another example of an edge-weighted subgraph detection problem with an exact requirement on the weight of the target subgraph is EXACT-EDGE-WEIGHT PERFECT MATCHING, which can be solved using algebraic techniques [22, §6] and has been used as a subroutine in subgraph isomorphism algorithms [21, Proposition 3.1].

The unweighted version of EEWC, obtained by setting all edge weights to 1, is NP-complete because it is equivalent to the CLIQUE problem. When using the number of vertices n as the complexity parameter, the problem admits a kernelization of size $\mathcal{O}(n^2)$ obtained by simply encoding the instance via its adjacency matrix. We prove the following lower bound, showing that the kernelization complexity of the edge-weighted version is a factor n larger. The lower bound even holds against *generalized* kernelizations (Definition 4).

► **Theorem 1.** *The EXACT-EDGE-WEIGHT CLIQUE problem parameterized by the number of vertices n does not admit a generalized kernelization of $\mathcal{O}(n^{3-\varepsilon})$ bits for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.*

Intuitively, the lower bound exploits the fact that the weight value of each of the $\Theta(n^2)$ edges in the instance may be a large integer requiring $\Omega(n)$ bits to encode. We also provide a randomized kernelization which matches this lower bound.

► **Theorem 2.** *There is a randomized polynomial-time algorithm that, given an n -vertex instance (G, w, t) of EXACT-EDGE-WEIGHT CLIQUE, outputs an instance (G', w', t') of bitsize $\mathcal{O}(n^3)$, in which each number is bounded by $2^{\mathcal{O}(n)}$, that is equivalent to (G, w, t) with probability at least $1 - 2^{-n}$. Moreover, if the input is a YES-instance, then the output is always a YES-instance.*

The proof is based on the idea that taking the weight function modulo a random prime preserves the answer to the instance with high probability. We adapt the argument by Harnik and Naor [13] that it suffices to pick a prime of magnitude $2^{\mathcal{O}(n)}$. As a result, each weight can be encoded with just $\mathcal{O}(n)$ bits.

It is noteworthy that the algorithm above can produce only false positives, therefore instead of using randomization we can turn it into a co-nondeterministic algorithm which guesses the correct values of the random bits. The framework of cross-composition excludes not only deterministic kernelization, but also co-nondeterministic [8], thus the lower bound from Theorem 1 indeed makes the presented algorithm tight.

Together, Theorems 1 and 2 pin down the kernelization complexity of EXACT-EDGE-WEIGHT CLIQUE, and prove it to be a factor n larger than for the unit-weight case. For CLIQUE, the kernelization of $\mathcal{O}(n^2)$ bits due to adjacency-matrix encoding cannot be improved to $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, as was shown by Dell and van Melkebeek [8].

We extend our results to the hypergraph setting, which is defined as follows: given a d -regular hypergraph ($d \geq 3$) with non-negative integer weights on the hyperedges, and a target value t , test if there is a vertex set S for which each size- d subset is a hyperedge (so

that S is a hyperclique) such that the sum of the weights of the hyperedges contained in S is exactly t . By a bootstrapping reduction using Theorem 1, we prove that EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE does not admit a generalized kernel of size $\mathcal{O}(n^{d+1-\varepsilon})$ for any $\varepsilon > 0$ unless $\text{NP} \subseteq \text{coNP}/\text{poly}$, while the randomized hashing technique yields a randomized kernelization of size $\mathcal{O}(n^{d+1})$.

We can view the edge-weighted (d -hyper)clique problem on (G, k, w, t) as a weighted constraint satisfaction problem (CSP) with weights from \mathbb{Z} , by introducing a binary variable for each vertex, and a weighted constraint for each subset S' of d vertices, which is satisfied precisely when all variables for S' are set to true. If S' is a (hyper)edge $e \in E(G)$ then the weight of the constraint on S' equals the weight of e ; if S' is not a hyperedge of G , then the weight of the constraint on S' is set to $-\infty$ to prevent all its vertices from being simultaneously chosen. Under this definition, G has a (hyper)clique of edge-weight t if and only if there is an assignment to the variables for which the total weight of satisfied constraints is t . Via this interpretation, the lower bounds for EEWC yield lower bounds on the kernelization complexity of weighted variants of CSP. We employ a recently introduced framework [17] of reductions among different CSPs whose constraint languages have the same maximum degree d of their characteristic polynomials, to transfer these lower bounds to other CSPs (see Section 3.3 for definitions). We obtain tight kernel bounds when parameterizing the exact-satisfaction-weight version of CSP by the number of variables, again using random prime numbers to obtain upper bounds. Our lower bounds for EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE transfer to all CSPs with degree $d \geq 2$. In degree-1 CSP each constraint depends on exactly one variable, therefore its exact-weighted variant is equivalent to the SUBSET SUM problem, for which we also provide a tight lower bound.

► **Theorem 3.** *SUBSET SUM parameterized by the number of items n does not admit a generalized kernelization of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.*

Theorem 3 tightens a result of Etscheid et al. [10, Theorem 14], who ruled out (standard) kernelizations for SUBSET SUM of size $\mathcal{O}(n^{2-\varepsilon})$ assuming the Exponential Time Hypothesis. Our reduction, conditioned on the incomparable assumption $\text{NP} \not\subseteq \text{coNP}/\text{poly}$, additionally rules out generalized kernelizations that compress into an instance of a potentially different problem. Note that the new lower bound implies that the input data in SUBSET SUM cannot be efficiently encoded in a more compact way, whereas the previous lower bound relies on the particular way the input is encoded in the natural formulation of the problem. On the other hand, a randomized kernel of size $\mathcal{O}(n^2)$ is known [13].

The results described so far characterize the kernelization complexity of broad classes of weighted constraint satisfaction problems in which the goal is to find a solution for which the total weight of satisfied constraints is exactly equal to a prescribed value. We also broaden our scope and investigate the maximization or minimization setting, in which the question is whether there is a solution whose cost is at least, or at most, a prescribed value. Some of our upper-bound techniques can be adapted to this setting: using a procedure by Nederlof, van Leeuwen and de Zwaan [24] a maximization problem can be reduced to a polynomial number of exact queries. This leads, for example, to a *Turing* kernelization (cf. [11]) for the weight-maximization version of d -UNIFORM HYPERCLIQUE which decides an instance in randomized polynomial time using queries of size $\mathcal{O}(n^{d+1})$ to an oracle for an auxiliary problem. We do not have lower bounds in the maximization regime.

In an attempt to understand the relative difficulty of obtaining an exact target weight versus maximizing the target weight, we finally investigate different models of weight reduction for the WEIGHTED VERTEX COVER problem studied extensively in earlier works [5, 10, 24]. We consider the problem on *bipartite* graphs, where an optimal solution can be found in

polynomial time, but we investigate whether a weight function can be efficiently compressed while either preserving (a) the collection of minimum-weight vertex covers, or (b) the relative ordering of total weight for all *inclusion-minimal* vertex covers. We give a polynomial-time algorithm for case (a) which reduces to a weight function with range $\{1, \dots, n\}$ using a relation to b -matchings, but show that in general it is impossible to achieve (b) with a weight function with range $\{1, \dots, 2^{o(n)}\}$, by utilizing lower bounds on the number of different threshold functions.

Organization. We begin with short preliminaries with the crucial definitions. We prove our main Theorem 1 in Section 3 by presenting a cross-composition of degree 3 into EXACT-EDGE-WEIGHT CLIQUE and employing it to obtain kernelization lower bounds for d -uniform hypergraphs for $d \geq 2$. This section also contains the kernelization lower bound for SUBSET SUM as well as the generalization of these results to Boolean CSPs. Next, in Section 4 we focus on bipartite WEIGHTED VERTEX COVER and the difficulty of compressing weight functions. The proofs of statements marked with (★) are located in the appendix. The proofs of statements marked with (♠) can be found in the full version [16]. The proof of Theorem 2, together with Turing kernelization for maximization problems, is given in Appendix B. The kernel upper bounds for Boolean CSPs can be found in the full version [16].

2 Preliminaries

We denote the set of natural numbers including zero by \mathbb{N}_0 , and the set of positive natural numbers by \mathbb{N}_+ . For positive integers n we define $[n] := \{1, \dots, n\}$. For a set U and integer $d \geq 1$ we denote by $\binom{U}{d}$ the collection of all size- d subsets of U . All logarithms we employ have base 2. Given a set U and a weight function $w: U \rightarrow \mathbb{N}_0$, for a subset $S \subseteq U$ we denote $w(S) := \sum_{v \in S} w(v)$.

A graph G has a vertex set $V(G)$ and an edge set $E(G) \subseteq \binom{V(G)}{2}$. For $d \geq 2$, a d -uniform hypergraph G consists of a vertex set $V(G)$ and a set of hyperedges $E(G) \subseteq \binom{V(G)}{d}$, that is, each hyperedge is a set of exactly d vertices. Hence a 2-uniform hypergraph is equivalent to a standard graph. A *clique* in a d -uniform hypergraph G is a vertex set $S \subseteq V(G)$ such that for each $X \in \binom{S}{d}$ we have $X \in E(G)$: each possible hyperedge among the vertices of S is present. A *vertex cover* for a graph G is a vertex set $S \subseteq V(G)$ containing at least one endpoint of each edge. A vertex cover is *inclusion-minimal* if no proper subset is a vertex cover.

Parameterized complexity. A *parameterized problem* Q is a subset of $\Sigma^* \times \mathbb{N}_+$, where Σ is a finite alphabet.

► **Definition 4.** Let $Q, Q' \subseteq \Sigma^* \times \mathbb{N}_+$ be parameterized problems and let $h: \mathbb{N}_+ \rightarrow \mathbb{N}_+$ be a computable function. A generalized kernel for Q into Q' of size $h(k)$ is an algorithm that, on input $(x, k) \in \Sigma^* \times \mathbb{N}_+$, takes time polynomial in $|x| + k$ and outputs an instance (x', k') such that:

1. $|x'|$ and k' are bounded by $h(k)$, and
2. $(x', k') \in Q'$ if and only if $(x, k) \in Q$.

The algorithm is a kernel for Q if $Q = Q'$. It is a polynomial (generalized) kernel if $h(k)$ is a polynomial.

► **Definition 5 (Linear-parameter transformations).** Let P and Q be parameterized problems. We say that P is linear-parameter transformable to Q , if there exists a polynomial-time computable function $f: \Sigma^* \times \mathbb{N}_+ \rightarrow \Sigma^* \times \mathbb{N}_+$, such that for all $(x, k) \in \Sigma^* \times \mathbb{N}_+$, (a) $(x, k) \in P$ if and only if $(x', k') = f(x, k) \in Q$ and (b) $k' \leq \mathcal{O}(k)$. The function f is called a linear-parameter transformation.

We employ a linear-parameter transformation for proving the lower bound for SUBSET SUM. For other lower bounds we use the framework of cross-composition [4] directly.

► **Definition 6** (Polynomial equivalence relation, [4, Def. 3.1]). *Given an alphabet Σ , an equivalence relation \mathcal{R} on Σ^* is called a polynomial equivalence relation if the following conditions hold.*

- (i) *There is an algorithm that, given two strings $x, y \in \Sigma^*$, decides whether x and y belong to the same equivalence class in time polynomial in $|x| + |y|$.*
- (ii) *For any finite set $S \subseteq \Sigma^*$ the equivalence relation \mathcal{R} partitions the elements of S into a number of classes that is polynomially bounded in the size of the largest element of S .*

► **Definition 7** (Degree- d cross-composition). *Let $L \subseteq \Sigma^*$ be a language, let \mathcal{R} be a polynomial equivalence relation on Σ^* , and let $Q \subseteq \Sigma^* \times \mathbb{N}_+$ be a parameterized problem. A degree- d OR-cross-composition of L into Q with respect to \mathcal{R} is an algorithm that, given z instances $x_1, x_2, \dots, x_z \in \Sigma^*$ of L belonging to the same equivalence class of \mathcal{R} , takes time polynomial in $\sum_{i=1}^z |x_i|$ and outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}_+$ such that:*

- (i) *the parameter k' is bounded by $\mathcal{O}(z^{1/d} \cdot (\max_i |x_i|)^c)$, where c is some constant independent of z , and*
- (ii) *$(x', k') \in Q$ if and only if there is an $i \in [z]$ such that $x_i \in L$.*

► **Theorem 8** ([4, Theorem 3.8]). *Let $L \subseteq \Sigma^*$ be a language that is NP-hard under Karp reductions, let $Q \subseteq \Sigma^* \times \mathbb{N}_+$ be a parameterized problem, and let $\varepsilon > 0$ be a real number. If L has a degree- d OR-cross-composition into Q and Q parameterized by k has a polynomial (generalized) kernelization of bitsize $\mathcal{O}(k^{d-\varepsilon})$, then $\text{NP} \subseteq \text{coNP/poly}$.*

3 Kernel lower bounds

3.1 Exact-Edge-Weight Clique

In this section we show that EXACT-EDGE-WEIGHT CLIQUE parameterized by the number of vertices in the given graph n does not admit a generalized kernel of size $\mathcal{O}(n^{3-\varepsilon})$, unless $\text{NP} \subseteq \text{coNP/poly}$. We use the framework of cross-composition to establish a kernelization lower bound [4]. We will use the NP-hard RED-BLUE DOMINATING SET (RBDS) as a starting problem for the cross-composition. Observe that RBDS is NP-hard because it is equivalent to SET COVER and HITTING SET [19].

RED-BLUE DOMINATING SET (RBDS)

Input: A bipartite graph G with a bipartition of $V(G)$ into sets R (red vertices) and B (blue vertices), and a positive integer $d \leq |R|$.

Question: Does there exist a set $D \subseteq R$ with $|D| \leq d$ such that every vertex in B has at least one neighbor in D ?

The following lemma forms the heart of the lower bound. It shows that an instance of EEWC on $z \cdot N^{\mathcal{O}(1)}$ vertices can encode the logical OR of a sequence of z^3 instances of size N each. Roughly speaking, this should be interpreted as follows: when $z \gg N$, each of the roughly z^2 edge weights of the constructed graph encodes z useful bits of information, in order to allow the instance on $\approx z^2$ edges to represent all z^3 inputs.

► **Lemma 9.** *There is a polynomial-time algorithm that, given integers z, d, n, m and a set of z^3 instances $\{(G_{i,j,k}, R_{i,j,k}, B_{i,j,k}, d) \mid i, j, k \in [z]\}$ of RBDS such that $|R_{i,j,k}| = m$ and $|B_{i,j,k}| = n$ for each $i, j, k \in [z]$, constructs an undirected graph G' , integer $t > 0$, and weight function $w: E(G') \rightarrow \mathbb{N}_0$ such that:*

1. the graph G' contains a clique of total edge-weight exactly t if and only if there exist $i^*, j^*, k^* \in [z]$ such that G_{i^*, j^*, k^*} has a red-blue dominating set of size at most d ,
2. the number of vertices in G' is $\mathcal{O}(z(m + nd))$, and
3. the values of t and $|V(G')|$ depend only on z, d, n , and m .

Proof. We describe the construction of (G', w, t) ; it will be easy to see that it can be carried out in polynomial time. Label the vertices in each set $R_{i,j,k}$ arbitrarily as r_1, \dots, r_m , and similarly label the vertices in each set $B_{i,j,k}$ as b_1, \dots, b_n . We construct a graph G' with edge-weight function w and integer t such that G' has a clique of total edge weight exactly t if and only if some $G_{i,j,k}$ is a YES-instance of RBDS. In the following construction we interpret edge weights as vectors of length $nz + 1$ written in base $(m + d + 2)$, which will be converted to integers later. Starting from an empty graph, we construct G' as follows; see Figure 1.

1. For each $i \in [z]$, create a vertex b_i . The vertices b_i form an independent set, so that any clique in G' contains at most one vertex b_i .
2. For each $j \in [z]$, create a vertex set $R_j = \{r_1^j, r_2^j, \dots, r_m^j\}$ and insert edges of weight $\vec{0}$ between all possible pairs of R_j .
3. For each $k \in [z]$, create a vertex s_k . The vertices s_k form an independent set, so that any clique in G' contains at most one vertex s_k .
4. For each $j, k \in [z]$, for each $x \in [m]$, insert an edge between s_k and r_x^j of weight $\vec{0}$.

The next step is to ensure that the neighborhood of a vertex r_x in $G_{i,j,k}$ is captured in the weights of the edges which are incident on r_x^j in G' .

5. For each $i, j \in [z]$, for each $x \in [m]$, insert an edge between b_i and r_x^j .
6. The weight of each edge $\{b_i, r_x^j\}$ is a vector of length $nz + 1$, out of which the least significant nz positions are divided into z blocks of length n each, and the most significant position is 1. The numbering of blocks as well as positions within a given block start with the least significant position.

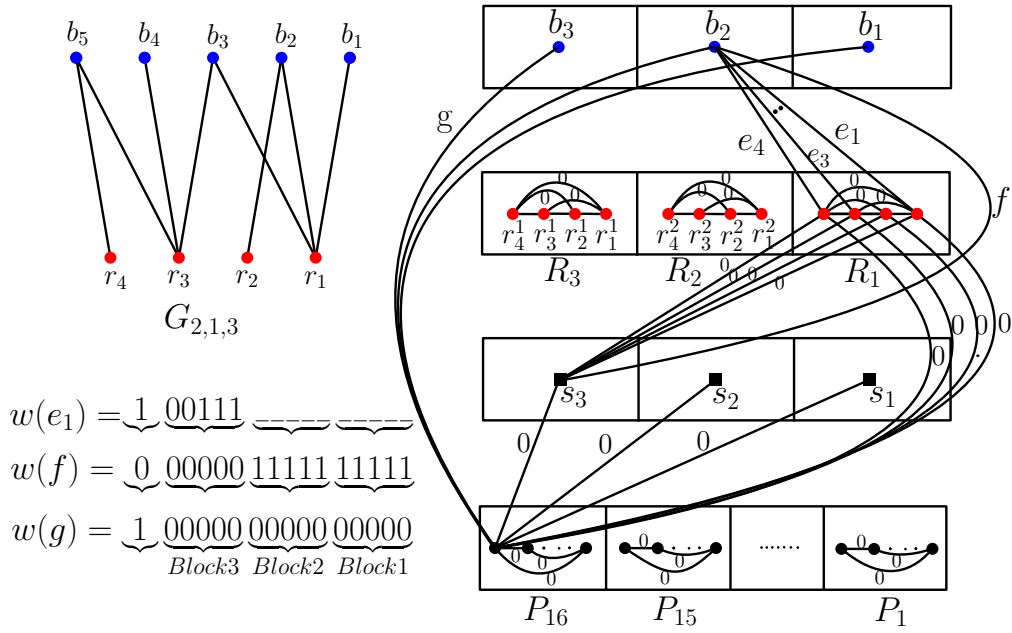
For each $i, j \in [z]$, for each $x \in [m]$, the weight of edge $\{b_i, r_x^j\}$ is defined as follows. For each $k \in [z]$, for each $q \in [n]$, the value $v_{k,q}(b_i, r_x^j)$ represents the value of the q^{th} position of the k^{th} block of the weight of $\{b_i, r_x^j\}$. The value is defined based on the neighborhood of vertex r_x in $G_{i,j,k}$ as follows:

$$v_{k,q}(b_i, r_x^j) = \begin{cases} 1 & \text{if } \{b_i, r_x\} \in E(G_{i,j,k}) \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Intuitively, the vector representing the weight of edge $\{b_i, r_x^j\}$ is formed by a 1 followed by the concatenation of z blocks of length n , such that the k^{th} block is the 0/1-incidence vector describing which of the n blue vertices of instance $G_{i,j,k}$ are adjacent to r_x .

Note that the n blue vertices of an input instance $G_{i,j,k}$ are represented by a single blue vertex b_i in G' . The difference between distinct blue vertices is encoded via different positions of the weight vectors. The most significant position of the weight vectors, which is always set to 1 for edges of the form $\{b_i, r_x^j\}$, will be used to keep track of the number of red vertices in a solution to RBDS.

The graph constructed so far has a mechanism to select the first index i of an instance $G_{i,j,k}$ (by choosing a vertex b_i), to select the second index j (by choosing vertices r_x^j), and to select the third index k (by choosing a vertex s_k). The next step in the construction adds weighted



■ **Figure 1** Top-left: An instance $(G_{2,1,3}, R_{2,1,3}, B_{2,1,3}, 2)$ of RBDS with $m = 4, n = 5$, and $d = 4$. Right: Illustration of the EEWC instance created for a sequence of 3^3 inputs including the one on the left. For readability, only a subset of the edges is drawn. Bottom-left: For each type edge with non-zero weight, an example weight is shown in vector form.

edges $\{b_i, s_k\}$, of which a solution clique in G' will contain exactly one. The weight vector for this edge is chosen so that the domination requirements from all RBDS instances whose third index differs from k (and which are therefore not selected) can be satisfied “for free”.

7. For each $i, k \in [z]$, insert an edge between b_i and s_k .
8. As in Step 6, the weight of the edge $\{b_i, s_k\}$ is a $(1 + nz)$ -tuple consisting of the most significant position followed by z blocks of length n . There is a 0 at the most significant position, block k consists of n zeros, and the other blocks are filled with ones. Hence the weight of the edge $\{b_i, s_k\}$ is independent of i .

To be able to ensure that G' has a clique of exactly weight t if some input instance $G_{i,j,k}$ has a solution, we need to introduce padding numbers which may be used as part of the solution to EEWC.

9. For each position $v \in [nz + 1]$ of a weight vector, add a vertex set $P_v = \{p_1^v, p_2^v, \dots, p_{d-1}^v\}$ to G' . Recall that d is the upper bound on the solution size for RBDS.
10. For each $i \in [z]$, for each $v \in [nz + 1]$, for each $y \in [d - 1]$, add an edge $\{b_i, p_y^v\}$. The weight of edge $\{b_i, p_y^v\}$ has value 1 at the v^{th} position and zeros elsewhere.
11. For each $v \in [nz + 1]$, for each $y \in [d - 1]$, add an edge $\{p_y^v, u\}$ of weight $\vec{0}$ for all $u \in V(G') \setminus (\{b_i \mid i \in [z]\} \cup \{p_y^v\})$, i.e., for all vertices $u \neq p_y^v$ which were not already adjacent to p_y^v .

We define the target weight t to be the $(nz + 1)$ -length vector with value d at each position, which satisfies Condition 3. Observe that G' has $\mathcal{O}(z(m + nd))$ vertices: Steps 1 and 3 contribute $\mathcal{O}(z)$ vertices, Step 2 contributes $\mathcal{O}(zm)$, and Step 9 contributes $\mathcal{O}(d(nz))$. Hence Condition 2 is satisfied. It remains to verify that G' has a clique of total edge weight

exactly t if and only if some input instance $G_{i,j,k}$ has a solution of RED-BLUE DOMINATING SET of size at most d . Before proving this property, we show the following claim which implies that no carries occur when summing up the weights of the edges of a clique in G' .

▷ **Claim 10.** For any clique $S \subseteq V(G')$, for any position $v \in [nz + 1]$ of a weight vector, there are at most $d + m + 1$ edges of the clique $G'[S]$ whose weight vector has a 1 at position v , and all other weight vectors are 0 at position v .

Proof. By construction, the entries of the vector encoding an edge weight are either 0 or 1.

By Steps 1 and 3, a clique S in G' contains at most one vertex b_i and one vertex s_k . Since G' does not have edges between vertices in distinct sets R_j and $R_{j'}$ by Step 2, any clique in G' consists of at most one vertex b_i , one vertex s_k , a subset of one set R_j , and a subset of $\bigcup_{v \in [nz+1]} P_v$. For any fixed position $v \in [nz + 1]$, the only edge-weight vectors which can have a 1 at position v are the $d - 1$ edges from P_v to b_i , the edge $\{b_i, s_k\}$, and the m edges between R_j and b_i . As this yields $(d - 1) + 1 + m$ edges that possibly have a 1 at position v , the claim follows. ◁

The preceding claim shows that when we convert each edge-weight vector to an integer by interpreting the vector as its base- $(m + d + 2)$ -representation, then no carries occur when computing the sum of the edge-weights of a clique. Hence the integer edge-weights of a clique $S \subseteq V(G')$ sum to the integer represented by vector t , if and only if the edge-weight vectors of the edges in S sum to the vector t . In the remainder, it therefore suffices to prove that there is a YES-instance G_{i^*,j^*,k^*} of RBDS among the inputs if and only if G' has a clique whose edge-weight vectors sum to the vector t . We prove these two implications.

▷ **Claim 11.** If some input graph G_{i^*,j^*,k^*} has a red-blue dominating set of size at most d , then G' has a clique of edge-weight exactly t .

Proof. Let $S \subseteq R_{i^*,j^*,k^*}$ of size at most d be a dominating set of B_{i^*,j^*,k^*} . We define a vertex set $S' \subseteq V(G')$ as follows. Initialize $S' := \{b_{i^*}, s_{k^*}\}$, and for each vertex $r_x \in S$, add the corresponding vertex $r_x^{j^*} \in R_{j^*}$ to S' .

We claim that S' is a clique in G' . To see this, note that R_{j^*} is a clique by Step 2. Vertex s_{k^*} is adjacent to all vertices of R_{j^*} by Step 4. Vertex b_{i^*} is adjacent to all vertices of R_{j^*} by Step 5. By Step 8 there is an edge between b_{i^*} and s_{k^*} .

Let us consider the weight of clique S' . Since S is a dominating set of B_{i^*,j^*,k^*} , if we sum up the weight vectors of the edges $\{b_{i^*}, r_x^{j^*}\}$ for $r_x \in S$, then by Step 6 we get a value of at least one at each position of block k^* . The most significant position of the resulting sum vector has value $|S| \leq d$. By Step 8 the weight vector of the edge $\{b_{i^*}, s_{k^*}\}$ consists of all ones, except for block k^* and the most significant position, where the value is zero. Thus adding the edge weight of $\{b_{i^*}, s_{k^*}\}$ to the previous sum ensures that each block has value at least 1 everywhere, whereas the most significant position has value $|S|$. All other edges spanned by S have weight $\vec{0}$. Letting t' denote the vector obtained by summing the weights of the edges of clique S' , we therefore find that t' has value $|S|$ as its most significant position and value at least 1 everywhere else.

Next we add some additional vertices to the set S' to get a clique of weight exactly t . By Step 11, vertices from the sets P_v for $v \in [nz + 1]$ are adjacent to all other vertices in the graph and can be added to any clique. All edges incident on a vertex $p_y^v \in P_v$ have weight $\vec{0}$, except the edges to vertices of the form b_i whose weight vector has a 1 at the v^{th} position and 0 elsewhere. Since S' contains exactly one such vertex b_{i^*} , for any $v \in [nz + 1]$ we can add up to $d - 1$ vertices from P_v to increase the weight sum at position v from its value of at least 1 in t' , to a value of exactly d . Hence G' has a clique of edge-weight exactly t . ◁

▷ **Claim 12.** If G' has a clique of edge-weight exactly t , then some input graph G_{i^*,j^*,k^*} has a red-blue dominating set of size at most d .

Proof. Suppose $G'[S']$ is a clique whose total edge weight is exactly t . Note that only edges for which one of the endpoints is of the form b_i for $i \in [z]$ have positive edge weights. The remaining edges all have weight $\vec{0}$. Also, by Step 1 there is at most one b -vertex in S' . Hence since $t \neq \vec{0}$ there is exactly one vertex b_{i^*} in S' . By Step 9 and 10, the edges of type $\{b_{i^*}, p_y^v\}$ for $p_y^v \in P_v$ contribute at most $d-1$ to the value of each position $v \in [nz+1]$ of the sum. Hence for each position $v \in [nz+1]$ there is an edge in clique S' of the form $\{b_{i^*}, r_x^j\}$ or $\{b_{i^*}, s_k\}$ which has a 1 at position v . We use this to show there is an input instance with a red-blue dominating set of size at most d .

By Step 3, there is at most one s -vertex in S' . Let $k^* := 1$ if $S \cap \{s_1, \dots, s_z\} = \emptyset$, and otherwise let s_{k^*} be the unique s -vertex in S' . Since the weight of the edge $\{b_{i^*}, s_{k^*}\}$ has zeros in block k^* by Step 8, our previous argument implies that for each of the n positions of block k^* , there is an edge in clique S' of the form $\{b_{i^*}, r_x^j\}$ whose weight has a 1 at that position. Hence S' contains at least one r -vertex, and by Step 2 all r -vertices in the clique S' are contained in a single set R_{j^*} . We show that G_{i^*,j^*,k^*} has a red-blue dominating set of size at most d . Let $S := \{r_x | r_x^j \in S'\}$. Since for each of the n positions of block k^* there is an edge $\{b_{i^*}, r_x^j\}$ in S' with a 1 at that position, by Step 5 each blue vertex of B_{i^*,j^*,k^*} has a neighbor in S . Hence S is a red-blue dominating set. By Step 5, the most significant position of each edge between b_{i^*} and R_{j^*} has value 1. As the most significant position of the target t is set to d , it follows that $|S| \leq d$, which proves that G_{i^*,j^*,k^*} has a red-blue dominating set of size at most d . ◁

This completes the proof of Lemma 9. ◀

Lemma 9 forms the main ingredient in a cross-composition that proves kernelization lower bounds for EXACT-EDGE-WEIGHT CLIQUE and its generalization to hypergraphs. For completeness, we formally define the hypergraph version as follows.

EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE (EEW- d -HC)
Input: A d -uniform hypergraph G , weight function $w: E(G) \rightarrow \mathbb{N}_0$, and a positive integer t .
Question: Does G have a hyperclique of total edge-weight exactly t ?

The following theorem generalizes Theorem 1. The case $d = 2$ of the theorem follows almost directly from Lemma 9 and Theorem 8, as the construction in the lemma gives the crucial ingredient for a degree-3 cross-composition. For larger d , we essentially exploit the fact that increasing the size of hyperedges by one allows one additional dimension of freedom, as has previously been exploited for other kernelization lower bounds for d -HITTING SET and d -SET COVER [7, 8]. The proof is given in Appendix A.1.

► **Theorem 13.** (★) *For each fixed $d \geq 2$, EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE parameterized by the number of vertices n does not admit a generalized kernel of size $\mathcal{O}(n^{d+1-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$.*

3.2 Subset Sum

We show that SUBSET SUM parameterized by the number of items n does not have generalized kernel of bitsize $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$. We prove the lower bound by giving a linear-parameter transformation from EXACT RED-BLUE DOMINATING SET. We

use EXACT RED-BLUE DOMINATING SET rather than RED-BLUE DOMINATING SET as our starting problem for this lower bound because it will simplify the construction: it will avoid the need for “padding” to cope with the fact that vertices are dominated multiple times.

The SUBSET SUM problem is formally defined as follows.

SUBSET SUM (SS) Input: A multiset X of n positive integers and a positive integer t . Question: Does there exist a subset $S \subseteq X$ with $\sum_{x \in S} x = t$?	Parameter: n
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------

We use the following problem as the starting point of the reduction.

EXACT RED-BLUE DOMINATING SET (ERBDS) Input: A bipartite graph G with a bipartition of $V(G)$ into sets R (red vertices) and B (blue vertices), and a positive integer $d \leq R $. Question: Does there exist a set $D \subseteq R$ of size <i>exactly</i> d such that every vertex in B has exactly one neighbor in D ?	Parameter: $n := V(G) $
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------

Jansen and Pieterse proved the following lower bound for ERBDS.

► **Theorem 14** ([15, Thm. 4.9]). *EXACT RED-BLUE DOMINATING SET parameterized by the number of vertices n does not admit a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.*

Actually, the lower bound they proved is for a slightly different variant of ERBDS where the solution D is required to have size *at most* d , instead of *exactly* d . Observe that the variant where we demand a solution of size *exactly* d is at least as hard as the *at most* d version: the latter reduces to the former by inserting d isolated red vertices. Therefore the lower bound by Jansen and Pieterse also works for the version we use here, which will simplify the presentation.

► **Theorem 3.** *SUBSET SUM parameterized by the number of items n does not admit a generalized kernelization of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.*

Proof. Given a graph G with a bipartition of $V(G)$ into R and B with $R = \{r_1, r_2, \dots, r_{n_R}\}$, $B = \{b_1, b_2, \dots, b_{n_B}\}$, and target value d for ERBDS, we transform it to an equivalent instance (X, t) of SS such that $|X| = n_R$. We start by defining n_R numbers N_1, N_2, \dots, N_{n_R} in base $(n_B + 1)$. For each $i \in [n_R]$, the number N_i consists of $(n_B + 1)$ digits. We denote the digits of the number N_i by $N_i[1], \dots, N_i[n_B + 1]$, where $N_i[1]$ is the least significant and $N_i[n_B + 1]$ is the most significant digit. Intuitively, the number N_i corresponds to the red vertex r_i . See Figure 2 for an illustration.

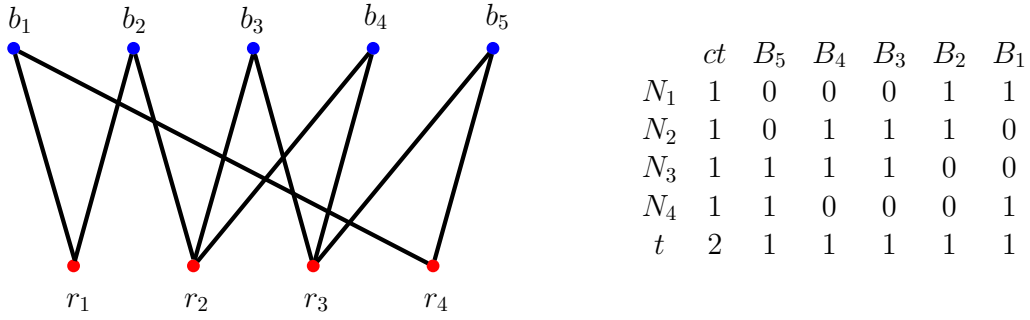
For each $i \in [n_R]$, for each $j \in [n_B + 1]$, digit $N_i[j]$ of number N_i is defined as follows:

$$N_i[j] = \begin{cases} 1 & \text{if } j = n_B + 1 \\ 1 & \text{if } j \in [n_B] \text{ and } \{r_i, b_j\} \in E(G) \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Hence the most significant digit of each number is 1, and the remaining digits of number N_i form the 0/1-vector indicating to which of the n_B blue vertices r_i is adjacent in G .

To complete the construction we set $X = \{N_1, N_2, \dots, N_{n_R}\}$ and we define t as follows:

$$t = d \underbrace{11 \dots 1}_{n_B \text{ times}} \quad (3)$$



■ **Figure 2** Left: An instance of ERBDS with $n_R = 4$, $n_B = 5$, and $d = 2$. Right: Illustration of the SS instance created for the given input. Note that $\{r_2, r_4\}$ and the numbers $\{N_2, N_4\}$ form a solution for ERBDS and SS, respectively. The leftmost column corresponds to the total count (ct) of the number of elements; the remaining columns correspond to blue vertices.

Observe that under these definitions, there are no carries when adding up a subset of the numbers in X , as each digit of each of the n_R numbers is either 0 or 1 and we work in base $n_R + 1$.

The number of items $|X|$ in the constructed instance of SS is n_R , upper bounded by the parameter $|V(G)|$ of ERBDS. It is easy to see that the construction can be carried out in polynomial time. To complete the linear-parameter transformation from ERBDS to SS, it remains to prove that G has a set $D \subseteq R$ of size exactly d such that every vertex in B has exactly one neighbor in D , if and only if there exist a set $S \subseteq X$ with $\sum_{x \in S} x = t$.

In the forward direction, suppose that there exists a set $D \subseteq R$ of size exactly d such that every vertex in B has exactly one neighbor in D . We claim that $\{N_i \mid r_i \in D\}$ is a solution to SS. The resulting sum has value d at the most significant digit since $|D| = d$. All other digits correspond to vertices in B . Since each blue vertex is adjacent to exactly one vertex from D it is easy to verify that all remaining digits of the sum are exactly one, implying that the numbers sum to exactly t .

For the reverse direction, suppose there is a set $S \subseteq X$ with $\sum_{x \in S} x = t$. Since the most significant digit of t is set to d and each number in X has a 1 as most significant digit, we have $|S| = d$ since there are no carries during addition. Define $D := \{r_i \mid N_i \in S\}$ as the set of the red vertices corresponding to the numbers in S . As $\sum_{x \in S} x = t$ and no carries occur in the summation, we have $\sum_{x \in S} x[j] = t[j] = 1$ for each $j \in [n_B]$. As the j -th digit of all numbers is either 0 or 1 by definition, there is a unique $N_i \in S$ with $N_i[j] = 1$, so that $r_i \in D$ is the unique neighbor of b_j in D . This shows that D is an exact red-blue dominating set of size d , concluding the linear-parameter transformation.

If there was a generalized kernelization for SS of size $\mathcal{O}(n^{2-\epsilon})$, then we would obtain a generalized kernelization for ERBDS of size $\mathcal{O}(n^{2-\epsilon})$ by first transforming it to SS, and then applying the generalized kernelization for the latter. Hence by contraposition and Theorem 14, the claim follows. ◀

3.3 Constraint Satisfaction Problems

In this section we extend our lower bounds to cover Boolean Constraint Satisfaction Problems (CSPs). We employ the recently introduced framework [17] of reductions among different CSPs to make a connection with EEW- d -HC. We start with introducing terminology necessary to identify crucial properties of CSPs.

Preliminaries on CSPs. A k -ary constraint is a function $f: \{0, 1\}^k \rightarrow \{0, 1\}$. We refer to k as the arity of f , denoted $\text{AR}(f)$. We always assume that the domain is Boolean. A constraint f is satisfied by an input $s \in \{0, 1\}^k$ if $f(s) = 1$. A constraint language Γ is a finite collection of constraints $\{f_1, f_2, \dots, f_\ell\}$, potentially with different arities. A *constraint application*, of a k -ary constraint f to a set of n Boolean variables, is a triple $\langle f, (i_1, i_2, \dots, i_k), w \rangle$, where the indices $i_j \in [n]$ select k of the n Boolean variables to whom the constraint is applied, and w is an integer weight. The variables can repeat in a single application.

A formula Φ of $\text{CSP}(\Gamma)$ is a set of constraint applications from Γ over a common set of variables. For an assignment x , that is, a mapping from the set of variables to $\{0, 1\}$, the integer $\Phi(x)$ is the sum of weights of the constraint applications satisfied by x . The considered decision problems are defined as follows.

<p>EXACT-WEIGHT $\text{CSP}(\Gamma)$</p> <p>Input: A formula Φ of $\text{CSP}(\Gamma)$ over n variables, an integer $t \in \mathbb{Z}$.</p> <p>Question: Is there an assignment x for which $\Phi(x) = t$?</p>	<p>Parameter: n</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------

<p>MAX-WEIGHT $\text{CSP}(\Gamma)$</p> <p>Input: A formula Φ of $\text{CSP}(\Gamma)$ over n variables, an integer $t \in \mathbb{Z}$.</p> <p>Question: Is there an assignment x for which $\Phi(x) \geq t$?</p>	<p>Parameter: n</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------

The compressibility of $\text{MAX-WEIGHT CSP}(\Gamma)$ has been studied by Jansen and Włodarczyk [17], who obtained essentially optimal kernel sizes for every Γ in the case where the weights are polynomial with respect to n . Even though the upper and lower bounds in [17] are formulated for $\text{MAX-WEIGHT CSP}(\Gamma)$, they could be adapted to work with $\text{EXACT-WEIGHT CSP}(\Gamma)$. The crucial idea which allows to determine compressibility of Γ is the representation of constraints via multilinear polynomials.

► **Definition 15.** For a k -ary constraint $f: \{0, 1\}^k \rightarrow \{0, 1\}$ its characteristic polynomial P_f is the unique k -ary multilinear polynomial over \mathbb{R} satisfying $f(x) = P_f(x)$ for any $x \in \{0, 1\}^k$.

It is known that such a polynomial always exists and it is unique [26].

► **Definition 16.** The degree of constraint language Γ , denoted $\text{deg}(\Gamma)$, is the maximal degree of a characteristic polynomial P_f over all $f \in \Gamma$.

The main result of Jansen and Włodarczyk [17] states that $\text{MAX-WEIGHT CSP}(\Gamma)$ with polynomial weights admits a kernel of $\mathcal{O}(n^{\text{deg}(\Gamma)} \log n)$ bits and, as long as the problem is NP-hard, it does not admit a kernel of size $\mathcal{O}(n^{\text{deg}(\Gamma)-\varepsilon})$, for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$. It turns out that in the variant when we allow both positive and negative weights the problem is NP-hard whenever $\text{deg}(\Gamma) \geq 2$ [18]. The lower bounds are obtained via linear-parameter transformations, where the parameter is the number of variables n . We shall take advantage of the fact that these transformations still work for an unbounded range of weights.

► **Lemma 17** ([17], Lemma 5.4). For constraint languages Γ_1, Γ_2 such that $2 \leq \text{deg}(\Gamma_1) \leq \text{deg}(\Gamma_2)$, there is a polynomial-time algorithm that, given a formula $\Phi_1 \in \text{CSP}(\Gamma_1)$ on n_1 variables and integer t_1 , returns a formula $\Phi_2 \in \text{CSP}(\Gamma_2)$ on n_2 variables and integer t_2 , such that

1. $n_2 = \mathcal{O}(n_1)$,
2. $\exists_x \Phi_1(x) = t_1 \iff \exists_y \Phi_2(y) = t_2$,
3. $\exists_x \Phi_1(x) \geq t_1 \iff \exists_y \Phi_2(y) \geq t_2$.

Kernel lower bounds for CSP. The lower bound of $\Omega(n^{\deg(\Gamma)-\varepsilon})$ has been obtained via a reduction from d -SAT (with $d = \deg(\Gamma)$) to MAX-WEIGHT CSP(Γ), combined with the fact that MAX d -SAT does not admit a kernel of size $\mathcal{O}(n^{d-\varepsilon})$ for $d \geq 2$ [8, 17]. We are going to show that when the weights are arbitrarily large, then the optimal compression size for EXACT-WEIGHT CSP(Γ) becomes essentially $\mathcal{O}(n^{\deg(\Gamma)+1})$, so the exponent is always larger by one compared to the case with polynomial weights. To this end, we are going to combine the aforementioned reduction framework with our lower bound for EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE.

Consider a constraint language Γ_{AND}^d consisting of a single d -ary constraint AND_d , which is satisfied only if all the arguments equal 1. The characteristic polynomial of AND_d is simply $P(x_1, \dots, x_d) = x_1 x_2 \cdots x_d$, hence the degree of Γ_{AND}^d equals d . We first translate our lower bounds for the hyperclique problems into a lower bound for EXACT-WEIGHT CSP(Γ_{AND}^d) for all $d \geq 2$, and then extend it to other CSPs.

► **Lemma 18.** *For all $d \geq 2$, EXACT-WEIGHT CSP(Γ_{AND}^d) does not admit a generalized kernel of size $\mathcal{O}(n^{d+1-\varepsilon})$, for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.*

Proof. Consider an instance (G, w, t) of EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE. Let W be the sum of all weights, which are by the definition non-negative. We can assume $t \in [0, W]$, as otherwise there is clearly no solution. We create an instance Φ of EXACT-WEIGHT CSP(Γ_{AND}^d) with the variable set $V(G)$ as follows. For each potential hyperedge $e = \{v_1, \dots, v_d\}$, if $e \in E(G)$ we create a constraint application $\langle \text{AND}_d, (v_1, \dots, v_d), w(e) \rangle$ and if $e \notin E(G)$, we create a constraint application $\langle \text{AND}_d, (v_1, \dots, v_d), W + 1 \rangle$.

If $X \subseteq V(G)$ is a hyperclique with total weight t , then for the assignment $x(v) = [v \in X]$ it holds that $\Phi(x) = t$. In the other direction, if $\Phi(x) = t$ then x cannot satisfy any constraint application with weight $W + 1$. Hence, each size- d subset of 1-valued variables corresponds to a hyperedge in G and $X = \{v \in V(G) \mid x(v) = 1\}$ forms a hyperclique of total weight t .

We have constructed a linear-parameter transformation from EEW- d -HC to EXACT-WEIGHT CSP(Γ_{AND}^d). Therefore, any generalized kernel of size $\mathcal{O}(n^{d+1-\varepsilon})$ for the latter would entail the same bound for EEW- d -HC. The claim follows from Theorem 13. ◀

The lower bound for EXACT-WEIGHT CSP(Γ_{AND}^d) given by Lemma 18 yields a lower bound for general EXACT-WEIGHT CSP(Γ) using the reduction framework described above.

► **Theorem 19.** *For any Γ with $\deg(\Gamma) \geq 2$, EXACT-WEIGHT CSP(Γ) does not admit a generalized kernel of size $\mathcal{O}(n^{\deg(\Gamma)+1-\varepsilon})$, for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.*

Proof. Consider an n -variable instance (Φ_1, t_1) of EXACT-WEIGHT CSP(Γ_{AND}^d), where $d = \deg(\Gamma)$. It holds that $\deg(\Gamma_{\text{AND}}^d) = d$. By Lemma 17, there is a linear-parameter transformation that translates (Φ_1, t_1) into an equivalent instance (Φ_2, t_2) of EXACT-WEIGHT CSP(Γ). If we could compress (Φ_2, t_2) into $\mathcal{O}(n^{d+1-\varepsilon})$ bits, this would entail the same compression for (Φ_1, t_1) . The claim follows from Lemma 18. ◀

This concludes the discussion of kernelization lower bounds. The kernelization upper bounds discussed in the introduction can be found in Appendix B (for hyperclique problems) and in the full version [16] (for CSPs).

4 Node-weighted Vertex Cover in bipartite graphs

Preserving all minimum solutions. For a graph G with node-weight function $w: V(G) \rightarrow \mathbb{N}_+$, we denote by $\mathcal{C}(G, w)$ the collection of subsets of $V(G)$ which are minimum-weight vertex covers of G . For n -vertex bipartite graphs there exists a weight function with range $[n]$ that preserves the set of minimum-weight vertex covers, which can be computed efficiently.

► **Theorem 20.** (♠) *There is an algorithm that, given an n -vertex bipartite graph G and node-weight function $w: V(G) \rightarrow \mathbb{N}_+$, outputs a weight function $w^*: V(G) \rightarrow [n]$ such that $\mathcal{C}(G, w) = \mathcal{C}(G, w^*)$. The running time of the algorithm is polynomial in $|V(G)|$ and the binary encoding size of w .*

The proof of the theorem is given in the full version [16]. It relies on the fact that a maximum b -matching (the linear-programming dual to VERTEX COVER) can be computed in strongly polynomial time in bipartite graphs by a reduction to MAX FLOW. The structure of a maximum b -matching allows two weight-reduction rules to be formulated whose exhaustive application yields the desired weight function. We also prove that the bound of n on the largest weight in Theorem 20 is best-possible.

Preserving the relative weight of solutions. For a graph G , we say that two node-weight functions w, w' are *vertex-cover equivalent* if the ordering of inclusion-minimal vertex covers by total weight is identical under the two weight functions, i.e., for all pairs of inclusion-minimal vertex covers $S_1, S_2 \subseteq V(G)$ we have $w(S_1) \leq w(S_2) \Leftrightarrow w'(S_1) \leq w'(S_2)$. While a minimum-weight vertex cover of a bipartite graph can be found efficiently, the following theorem shows that nevertheless weight functions with exponentially large coefficients may be needed to preserve the ordering of minimal vertex covers by weight.

► **Theorem 21.** (♠) *For each $n \geq 1$, there exists a node-weighted bipartite graph G_n on $2(n+1)$ vertices with weight function $w: V(G_n) \rightarrow \mathbb{N}_+$ such that for all weight functions $w': V(G) \rightarrow \mathbb{N}_+$ which are vertex-cover equivalent to w , we have: $\max_{v \in V(G_n)} w'(v) \geq 2^{\Omega(n)}$.*

5 Conclusions

We have established kernelization lower bounds for SUBSET SUM, EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE, and a family of EXACT-WEIGHT CSP problems, which make it unlikely that there exists an efficient algorithm to compress a single weight into $o(n)$ bits. This gives a clear separation between the setting involving arbitrarily large weights and the case with polynomially-bounded weights, which can be encoded with $\mathcal{O}(\log n)$ bits each. The matching kernel upper bounds are randomized and we leave it as an open question to derandomize them. For SUBSET SUM parameterized by the number of items n , a deterministic kernel of size $\mathcal{O}(n^4)$ is known [10].

Kernelization of weighted minimization/maximization problems is so far less understood. We are able to match the same kernel size as for the exact-weight problems, but only through Turing kernels. Using techniques from [10] one can obtain, e.g., a kernel of size $\mathcal{O}(n^8)$ for MAX-EDGE-WEIGHT CLIQUE. Improving upon this bound possibly requires a better understanding of the threshold functions. Our study of weighted VERTEX COVER on bipartite graphs indicates that preserving the order between all the solutions might be overly demanding and it could be easier to keep track only of the structure of the optimal solutions. Can we extend the theory of threshold functions so that better bounds are feasible when we just want to maintain a separation between optimal and non-optimal solutions?

References

- 1 Amir Abboud, Shon Feller, and Oren Weimann. On the fine-grained complexity of parity problems. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPICs*, pages 5:1–5:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.5.

- 2 Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2014. doi:10.1007/978-3-662-44777-2_1.
- 3 Andreas Björklund. Determinant sums for undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.
- 4 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014. doi:10.1137/120880240.
- 5 Miroslav Chlebík and Janka Chlebíková. Crown reductions for the minimum weighted vertex cover problem. *Discret. Appl. Math.*, 156(3):292–312, 2008. doi:10.1016/j.dam.2007.03.026.
- 6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 7 Holger Dell and Dániel Marx. Kernelization of packing problems. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 68–81. SIAM, 2012. doi:10.1137/1.9781611973099.6.
- 8 Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.
- 9 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
- 10 Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *J. Comput. Syst. Sci.*, 84:1–10, 2017. doi:10.1016/j.jcss.2016.06.004.
- 11 Henning Fernau. Kernelization, Turing kernels. In *Encyclopedia of Algorithms*, pages 1043–1045. Springer, 2016. doi:10.1007/978-1-4939-2864-4_528.
- 12 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.
- 13 Danny Harnik and Moni Naor. On the compressibility of NP instances and cryptographic applications. *SIAM Journal on Computing*, 39(5):1667–1713, 2010. doi:10.1137/060668092.
- 14 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978. doi:10.1137/0207033.
- 15 Bart M. P. Jansen and Astrid Pieterse. Optimal sparsification for some binary CSPs using low-degree polynomials. *TOCT*, 11(4):28:1–28:26, 2019. doi:10.1145/3349618.
- 16 Bart M. P. Jansen, Shivesh K. Roy, and Michał Włodarczyk. On the hardness of compressing weights, 2021. arXiv:2107.02554.
- 17 Bart M. P. Jansen and Michal Włodarczyk. Optimal polynomial-time compression for Boolean Max CSP. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 63:1–63:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ESA.2020.63.
- 18 Peter Jonsson and Andrei Krokhin. Maximum H -colourable subdigraphs and constraint optimization with arbitrary weights. *Journal of Computer and System Sciences*, 73(5):691–702, 2007. doi:10.1016/j.jcss.2007.02.001.
- 19 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.
- 20 Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *IBM journal of research and development*, 31(2):249–260, 1987.
- 21 Dániel Marx and Michal Pilipczuk. Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask), 2013. arXiv:1307.2187v3.

- 22 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. doi:10.1007/BF02579206.
- 23 Jesper Nederlof. Bipartite TSP in $o(1.9999^n)$ time, assuming quadratic time matrix multiplication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 40–53. ACM, 2020. doi:10.1145/3357713.3384264.
- 24 Jesper Nederlof, Erik Jan van Leeuwen, and Ruben van der Zwaan. Reducing a target interval to a few exact queries. In Branislav Rován, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012 – 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings*, volume 7464 of *Lecture Notes in Computer Science*, pages 718–727. Springer, 2012. doi:10.1007/978-3-642-32589-2_62.
- 25 G.L. Nemhauser and L.E.jun. Trotter. Vertex packings: structural properties and algorithms. *Math. Program.*, 8:232–248, 1975. doi:10.1007/BF01580444.
- 26 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994. doi:10.1007/BF01263419.
- 27 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPICs*, pages 17–29. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.IPEC.2015.17.
- 28 Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 455–464. ACM, 2009. doi:10.1145/1536414.1536477.
- 29 Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. doi:10.1137/09076619X.
- 30 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.

A Kernel lower bounds

A.1 Omitted proof for Exact-Edge-Weight Clique

► **Theorem 13.** (★) *For each fixed $d \geq 2$, EXACT-EDGE-WEIGHT d -UNIFORM HYPER-CLIQUE parameterized by the number of vertices n does not admit a generalized kernel of size $\mathcal{O}(n^{d+1-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$.*

Proof. We give a degree- $(d + 1)$ OR-cross-composition (Definition 7) from RBDS to the weighted hyperclique problem using Lemma 9. We start by giving a polynomial equivalence relation \mathcal{R} on inputs of RBDS. Let two instances of RBDS be equivalent under \mathcal{R} if they have the same number of red vertices, the same number of blue vertices, and the same target value d . It is easy to check that \mathcal{R} is a polynomial equivalence relation.

Consider Z inputs of RBDS from the same equivalence class of \mathcal{R} . If Z is not a $(d + 1)^{\text{th}}$ power of an integer, then we duplicate one of the input instances until we reach the first number of the form $2^{(d+1)i}$, which is trivially such a power. This increases the number of instances by at most the constant factor 2^{d+1} and does not change whether there is a YES-instance among the instances. As all requirements on a cross-composition are oblivious to constant factors, from now on we may assume without loss of generality that $Z = z^{d+1}$ for some integer z . By definition of \mathcal{R} , all instances have the same number m of red vertices, the same number n of blue vertices, and have the same maximum size d of a solution.

64:18 On the Hardness of Compressing Weights

For $d = 2$, we can simply invoke Lemma 9 for the $z^{d+1} = z^3$ instances of RBDS and output the resulting instance (G', w, t) of EEWC, which acts as the logical OR. Since the encoding size N of an instance of RBDS with m red vertices, n blue vertices, and target value d satisfies $N \in \Omega(n + m + d)$, Lemma 9 guarantees that G' has $\mathcal{O}(z(m + nd)) \in \mathcal{O}(\sqrt[3]{Z} \cdot N^2)$ vertices, which is suitably bounded for a degree-3 cross-composition for the parameterization by the number of vertices. Hence the claimed lower bound for generalized kernelization then follows from Theorem 8.

In the remainder of the proof, we assume $d \geq 3$. Partition the z^{d+1} inputs in z^{d-2} groups $\{X_{i_1, \dots, i_{d-2}} \mid i_1, \dots, i_{d-2} \in [z]\}$ of size z^3 each. Apply Lemma 9 to each group $X_{i_1, \dots, i_{d-2}}$. This results in z^{d-2} instances $(G_{i_1, \dots, i_{d-2}}, w_{i_1, \dots, i_{d-2}}, t)$ of EEWC on a simple graph. Note that all instances share the same value of $t > 0$, as Lemma 9 ensures that t only depends on (z, d, n, m) which are identical for all groups. Similarly, all resulting instances have the same number of vertices. Hence we can re-label the vertices in each graph so that all graphs $G_{i_1, \dots, i_{d-2}}$ have the same vertex set \mathcal{V} of size $\mathcal{O}(z(m + nd))$. The YES/NO-answer to each composed instance is the disjunction of the answers to the RBDS instances in its corresponding group.

Build a d -uniform hypergraph G^* with weight function $w^*: E(G^*) \rightarrow \mathbb{N}_0$ and target value t^* as follows:

1. $V(G^*) = \mathcal{V} \cup Y_1 \cup \dots \cup Y_{d-2}$, where $Y_\ell = \{y_{\ell, j} \mid j \in [z]\}$ for $\ell \in [d-2]$.
2. A set $S \subseteq V(G^*)$ of exactly d vertices is a hyperedge of G^* if there is no $\ell \in [d-2]$ for which $|S \cap Y_\ell| > 1$.
3. The weight of a hyperedge S is equal to 0 if there exists $\ell \in [d-2]$ with $S \cap Y_\ell = \emptyset$. Otherwise, for each $\ell \in [d-2]$ let i_ℓ be the unique index j such that $y_{\ell, j} \in S$.
 - If $e_S := S \cap \mathcal{V}$ is an edge in graph $G_{i_1, \dots, i_{d-2}}$ then define $w^*(S) := w_{i_1, \dots, i_{d-2}}(e_S)$.
 - Otherwise, let $w^*(S) := t + 1$.
4. Set $t^* = t$.

Since $d \in \mathcal{O}(1)$, hypergraph G^* has $\mathcal{O}(z \cdot (m + nd)) + \mathcal{O}(z \cdot d) \in \mathcal{O}(Z^{1/(d+1)} \cdot (m + n)^{\mathcal{O}(1)})$ vertices. (We use here that $d \leq m$.) Hence the parameter value of the constructed EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE instance is indeed bounded by the $(d+1)$ -th root of the number of input instances times a polynomial in the maximum size of an input instance, satisfying the parameter bound of a degree- $(d+1)$ cross-composition.

It remains to verify that G^* has a hyperclique of weight t^* if and only if one of the input instances has a RBDS of size at most d . By the guarantee of Lemma 9, it suffices to show that G^* has a hyperclique of weight t^* if and only if one of the weighted standard graphs $(G_{i_1, \dots, i_{d-2}}, w_{i_1, \dots, i_{d-2}})$ obtained by applying that lemma to some group of z^3 inputs, has a clique of weight t .

First suppose there exists a weighted graph $(G_{i_1^*, \dots, i_{d-2}^*}, w_{i_1^*, \dots, i_{d-2}^*})$ that contains a clique S of total edge weight t . Let $I := \{y_{\ell, i_\ell^*} \mid \ell \in [d-2]\}$. Let $S' := S \cup I$. By Step 2, the set S' is a hyperclique in G^* . It remains to verify that its weight is $t^* = t$. By Step 3, for each edge e of the clique S the set $e \cup I$ is a hyperedge in G^* of the same weight. Additionally, each subset of S' that does not contain I has weight 0. Hence the weight of hyperclique S' is equal to the weight of clique S and is therefore $t^* = t$.

For the other direction, suppose G^* has a clique $G^*[S^*]$ of weight $t^* = t$. Since $t > 0$ and all hyperedges in G^* of nonzero weight contain exactly one vertex of each set Y_ℓ for $\ell \in [d-2]$, there exist i_1^*, \dots, i_{d-2}^* such that $S^* \cap Y_\ell = \{i_\ell^*\}$ for each $\ell \in [d-2]$. Let $I := \{y_{\ell, i_\ell^*} \mid \ell \in [d-2]\}$. We will show that $S^* \cap \mathcal{V}$ is a clique of weight t in $G_{i_1^*, \dots, i_{d-2}^*}$. Since $t^* = t > 0$ and edge-weights are non-negative, it follows that no hyperedge in S^* has weight $t + 1$. By Step 3, this implies each subset of $S^* \cap \mathcal{V}$ of size two is an edge of $G_{i_1^*, \dots, i_{d-2}^*}$, and hence $S^* \cap \mathcal{V}$ is a clique.

For each set $e \subseteq \mathcal{V}$ of size two, the weight of the hyperedge $e \cup I$ is equal to $w_{i_1^*, \dots, i_{d-2}^*}(e)$. As all other hyperedges in S^* have weight 0, it follows that the weight of the clique $S^* \cap \mathcal{V}$ equals that of hyperclique S^* , and is therefore equal to $t^* = t$. This implies $G_{i_1^*, \dots, i_{d-2}^*}$ has a clique of total edge weight $t = t^*$, which concludes the proof. \blacktriangleleft

B Kernel upper bounds

In this section, we present randomized kernel upper bounds for EEW- d -HC, which match the obtained lower bounds. For the maximization variant of EEW- d -HC, we present a Turing kernel with the same bounds. The results in this section follow from combining known arguments from Harnik and Naor [13] and Nederlof et al. [23] with gadgets that allow us to produce an instance of the same problem that is being compressed (so we obtain a true kernelization, not a generalized one).

Consider a family \mathcal{F} of subsets of a universe U and a weight function $w: \mathcal{F} \rightarrow [-N, N]$. For a subset $X \subseteq U$, we denote $w_{\text{SUM}}(X) = \sum_{Y \in \mathcal{F}, Y \subseteq X} w(Y)$. The following fact has been observed by Harnik and Naor [13, Claim 2.7] and for the sake of completeness we provide a proof for the formulation which is the most convenient for us.

► **Lemma 22.** *Let U be a set of size n , $\mathcal{F} \subseteq 2^U$ be a family of subsets, $w: \mathcal{F} \rightarrow [-N, N]$ be a weight function, and $t \in [-N, N]$. There exists a randomized polynomial-time algorithm that, given a real $\varepsilon > 0$, returns a prime number $p \leq 2^n \cdot \text{poly}(n, \log N, \varepsilon^{-1})$, such that if there is no $X \subseteq U$ satisfying $w_{\text{SUM}}(X) = t$, then*

$$\mathbb{P}\left(\text{there is } X \subseteq U \text{ satisfying } w_{\text{SUM}}(X) \equiv t \pmod{p}\right) \leq \varepsilon.$$

Proof. For a fixed function w , we say that p is *bad* if for some $X \subseteq U$ it holds that $w_{\text{SUM}}(X) \equiv t \pmod{p}$ but $w_{\text{SUM}}(X) \neq t$. This implies that p divides $|w_{\text{SUM}}(X) - t|$. We argue that the number of bad primes is bounded by $2^n \cdot (n + 1 + \log(N))$. Since $|w_{\text{SUM}}(X) - t| \leq 2^{n+1} \cdot N$, this number can have at most $\log(2^{n+1} \cdot N) = n + 1 + \log N$ different prime divisors. There are at most 2^n choices of X , which proves the bound.

We sample a random prime p among the set of the first $M = 2^n \cdot (n + 1 + \log(N)) \cdot \varepsilon^{-1}$ primes. It is known that the first M primes lie in the interval $[2, \mathcal{O}(M \log M)]$ and we can uniformly sample a prime number from this interval in time $(\log M)^{\mathcal{O}(1)} = (n + \log \log(N) + \log(\varepsilon^{-1}))^{\mathcal{O}(1)}$ [20]. By the argument above, the probability of choosing a bad prime is bounded by ε . \blacktriangleleft

► **Theorem 23.** *There is a randomized polynomial-time algorithm that, given an n -vertex instance (G, w, t) of EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE, outputs an instance (G', w', t') of bitsize $\mathcal{O}(n^{d+1})$, such that:*

1. *if (G, w, t) is a YES-instance, then (G', w', t') is always a YES-instance,*
2. *if (G, w, t) is a NO-instance, then (G', w', t') is a NO-instance with probability at least $1 - 2^{-n}$.*

Furthermore, each number in (G', w', t') is bounded by $2^{\mathcal{O}(n)}$.

Proof. Let us define $N = \max(t, \max_{e \in E(G)} w_e)$. We can assume $\log N \leq 2^n$, because otherwise the input length is lower bounded by 2^n and the brute-force algorithm for EEW- d -HC becomes polynomial.

We apply Lemma 22 to the weight function w , target t , and $\varepsilon = 2^{-n}$, to compute the desired prime $p \leq 2^n \cdot \text{poly}(n, \log N, \varepsilon^{-1}) = 2^{\mathcal{O}(n)}$. If there exists a hyperclique $X \subseteq V(G)$ satisfying $w_{\text{SUM}}(X) = t$ with respect to the weighted set family $E(G) \subseteq \binom{V(G)}{d}$, then clearly $w_{\text{SUM}}(X) \equiv t \pmod{p}$. Furthermore, with probability $1 - 2^{-n}$, the implication in the other direction holds as well. In particular, in this case p does not divide t .

Let us construct a new instance (G', w', t') of EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE with weights bounded by $p \cdot n^d$, which is bounded by $2^{\mathcal{O}(n)}$ for constant d . We set $w'(v) = w(v) \pmod{p}$ and $t_p = t \pmod{p}$. The condition $w_{\text{SUM}}(X) \equiv t \pmod{p}$ is equivalent to the existence of $i \in [0, n^d)$ for which $w'_{\text{SUM}}(X) = t_p + ip$, because the sum $w'_{\text{SUM}}(X)$ comprises of at most n^d summands from the range $[0, p)$.

We introduce a set U_Z of $d - 1$ new vertices and for each $j \in [0, d - 1]$ we introduce a set U_j of n new vertices. Intuitively, the sets U_j can be used to represent any number $i \in [0, n^d)$ in base n . For every j and every $v \in U_j$ we create a hyperedge $e = U_Z \cup \{v\}$ with weight $w'_e = n^j \cdot p$. For every other size- d subset containing at least one new vertex, we create a hyperedge with weight 0. Observe that for every integer $i \in [0, n^d]$, we can find a set $Y \subseteq U_Z \cup U_0 \cup \dots \cup U_{d-1}$ such that $w'_{\text{SUM}}(Y) = ip$. Let G' be the graph with the set of vertices $V(G) \cup U_Z \cup U_0 \cup \dots \cup U_{d-1}$ and hyperedges inherited from G plus these defined above. We set $t' = t_p + n^d \cdot p$.

Suppose now that $X \subseteq V(G)$ forms a hyperclique of total weight t in G . Then $w'_{\text{SUM}}(X) = t_p + ip$ for some $i \in [0, n^d)$. By the argument above, we can find a set $Y \subseteq U_Z \cup U_0 \cup \dots \cup U_{d-1}$ such that $w'_{\text{SUM}}(X \cup Y) = t'$ and $X \cup Y$ is a hyperclique in G' .

In the other direction, suppose we have successfully applied Lemma 22 and there is a hyperclique $X' \subseteq V(G')$ with total weight t' . Then p divides $w'_{\text{SUM}}(X' \setminus V(G))$, so since all hyperedges intersecting both $V(G)$ and $X' \setminus V(G)$ have weight 0, we have $w'_{\text{SUM}}(X' \cap V(G)) \equiv t \pmod{p}$ and $w_{\text{SUM}}(X' \cap V(G)) = t$, which gives a desired hyperclique in G .

The new instance has $\mathcal{O}(n)$ vertices and $\mathcal{O}(n^d)$ edges. The weight range is $[0, n^d \cdot p]$ and, since $p = 2^{\mathcal{O}(n)}$, each weight can be encoded with $\mathcal{O}(n)$ bits. The claim follows. \blacktriangleleft

We obtain Theorem 2 as a corollary by taking $d = 2$.

B.1 Turing kernel for Max Weighted Hyperclique

We turn our attention to the maximization variant of the weighted hyperclique problem. We consider the problem MAX-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE, which takes the same input as EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE, but the goal is to detect a hyperclique of total weight greater or equal to the target value t . Even though we are not able to compress the weight function as in Theorem 23, we present a Turing kernelization with the same size. We rely on a generic technique of reducing interval queries to exact queries.

► **Theorem 24** ([24], Theorem 1). *Let U be a set of cardinality n , let $w: U \rightarrow \mathbb{N}_0$ be a weight function, and let $l < u$ be non-negative integers with $u - l > 1$. There is a polynomial-time algorithm that returns a set of pairs $\Omega = (w_1, t_1), \dots, (w_K, t_K)$ with $w_i: U \rightarrow \mathbb{N}_0$ and integers t_1, t_2, \dots, t_K , such that:*

1. K is at most $(5n + 2) \cdot \log(u - l)$,
2. for every set $X \subseteq U$ it holds that $w(X) \in [l, u]$ if and only if there exist $i \in [1, K]$ such that $w_i(X) = t_i$.

A polynomial Turing kernel (cf. [11]) for a parameterized problem \mathcal{P} is a polynomial-time algorithm that decides any instance of \mathcal{P} with access to an oracle that answers instances of size polynomial with respect to the parameter. The size of a Turing kernel is the maximal size of the instances queried to the oracle. Note that if the classic problem underlying \mathcal{P} is NP-hard, then any query for a potentially different problem \mathcal{Q} can be translated into a query for \mathcal{P} whose size is only polynomially larger. Hence in many settings, including ours, it does not make a difference for the existence of polynomial-size Turing kernels whether the queries are for the same problem or for another problem contained in NP.

Observe that the number of calls to the oracle is not restricted, although it is polynomial in the input size since the overall procedure runs in polynomial time. The following theorem gives a one-sided error randomized Turing kernel of size $\mathcal{O}(n^{d+1})$ for MAX-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE parameterized by the number of vertices n .

► **Theorem 25.** *There is a randomized polynomial-time algorithm that, given an n -vertex instance (G, w, t) of MAX-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE, returns a family of K instances (G_i, w_i, t_i) , $i \in [K]$, of EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE, each of bitsize $\mathcal{O}(n^{d+1})$, such that:*

1. K is polynomial with respect to the input size,
2. if (G, w, t) is a YES-instance, then at least one instance (G_i, w_i, t_i) is a YES-instance,
3. if (G, w, t) is a NO-instance, then with probability $1 - 2^{-\Omega(n)}$ all the instances (G_i, w_i, t_i) are NO-instances.

Proof. We apply Theorem 24 with U being the set of hyperedges in G , $l = t$, and $u = n^d \cdot \max_{e \in E(G)} w_e$. We can assume that $l \leq u$, as otherwise there can be no solution. We obtain $K = \log(u - l) \cdot \mathcal{O}(n^d)$ many weight functions w^i and integers t_i , so that for each $X \subseteq U$ it holds $w_{\text{SUM}}(X) \geq t$ if and only if $w_{\text{SUM}}^i(X) = t_i$ for some $i \in [K]$. Observe that $\log(u - l)$ is upper bounded by the input size, so the condition (1) is satisfied.

The original problem thus reduces to a disjunction of polynomially many instances of EXACT-EDGE-WEIGHT d -UNIFORM HYPERCLIQUE. We use Theorem 23 to compress each of them to $\mathcal{O}(n^{d+1})$ bits. The probability that a single instance would be incorrectly compressed is bounded by 2^{-n} . By the union bound, the probability that any instance would be incorrectly compressed is $n^{\mathcal{O}(1)} \cdot 2^{-n} = 2^{-\Omega(n)}$. ◀

Griddings of Permutations and Hardness of Pattern Matching

Vít Jelínek  

Computer Science Institute, Charles University, Prague, Czech Republic

Michal Opler  

Computer Science Institute, Charles University, Prague, Czech Republic

Jakub Pekárek  

Department of Applied Mathematics, Charles University, Prague, Czech Republic

Abstract

We study the complexity of the decision problem known as *Permutation Pattern Matching*, or PPM. The input of PPM consists of a pair of permutations τ (the “text”) and π (the “pattern”), and the goal is to decide whether τ contains π as a subpermutation. On general inputs, PPM is known to be NP-complete by a result of Bose, Buss and Lubiw. In this paper, we focus on restricted instances of PPM where the text is assumed to avoid a fixed (small) pattern σ ; this restriction is known as $\text{Av}(\sigma)$ -PPM. It has been previously shown that $\text{Av}(\sigma)$ -PPM is polynomial for any σ of size at most 3, while it is NP-hard for any σ containing a monotone subsequence of length four.

In this paper, we present a new hardness reduction which allows us to show, in a uniform way, that $\text{Av}(\sigma)$ -PPM is hard for every σ of size at least 6, for every σ of size 5 except the symmetry class of 41352, as well as for every σ symmetric to one of the three permutations 4321, 4312 and 4231. Moreover, assuming the exponential time hypothesis, none of these hard cases of $\text{Av}(\sigma)$ -PPM can be solved in time $2^{o(n/\log n)}$. Previously, such conditional lower bound was not known even for the unconstrained PPM problem.

On the tractability side, we combine the CSP approach of Guillemot and Marx with the structural results of Huczynska and Vatter to show that for any monotone-griddable permutation class \mathcal{C} , PPM is polynomial when the text is restricted to a permutation from \mathcal{C} .

2012 ACM Subject Classification Mathematics of computing \rightarrow Permutations and combinations; Theory of computation \rightarrow Pattern matching; Theory of computation \rightarrow Problems, reductions and completeness

Keywords and phrases Permutation, pattern matching, NP-hardness

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.65

Related Version *Full Version:* <https://arxiv.org/abs/2107.10897>

Funding Supported by the GAUK project 1766318.

Vít Jelínek: Supported by project 18-19158S of the Czech Science Foundation.

Michal Opler: Supported by project 21-32817S of the Czech Science Foundation.

1 Introduction

Permutation Pattern Matching, or PPM, is one of the most fundamental decision problems related to permutations. In PPM, the input consists of two permutations: τ , referred to as the “text”, and π , referred to as the “pattern”. The two permutations are represented as sequences of distinct integers. The goal is to determine whether the text τ contains the pattern σ , that is, whether τ has a subsequence order-isomorphic to σ (see Section 2 for precise definitions).

Bose, Buss and Lubiw [7] have shown that the PPM problem is NP-complete. Thus, most recent research into the complexity of PPM focuses either on parametrized or superpolynomial algorithms, or on restricted instances of the problem.



© Vít Jelínek, Michal Opler, and Jakub Pekárek;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 65; pp. 65:1–65:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

For a pattern π of size k and a text τ of size n , a straightforward brute-force approach can solve PPM in time $O(n^{k+1})$. This was improved by Ahal and Rabinovich [1] to $O(n^{0.47k+o(k)})$, and then by Berendsohn, Kozma and Marx [6] to $O(n^{k/4})$.

When k is large in terms of n , a brute-force approach solves PPM in time $O(2^{n+o(n)})$. The first improvement upon this bound was obtained by Bruner and Lackner [8], whose algorithm achieves the running time $O(1.79^n)$, which was in turn improved by Berendsohn, Kozma and Marx [6] to $O(1.6181^n)$.

Guillemot and Marx [11] have shown, perhaps surprisingly, that PPM is fixed-parameter tractable with parameter k , via an algorithm with running time $n \cdot 2^{O(k^2 \log k)}$, later improved to $n \cdot 2^{O(k^2)}$ by Fox [10].

Restricted instances

Given that PPM is NP-hard on general inputs, various authors have sought to identify restrictions on the input permutations that would allow for an efficient pattern matching algorithm. These restrictions usually take the form of specifying that the pattern must belong to a prescribed set \mathcal{C} of permutations (the so-called \mathcal{C} -PATTERN PPM problem), or that both the pattern and the text must belong to a set \mathcal{C} (known as \mathcal{C} -PPM problem). The most commonly considered choices for \mathcal{C} are sets of the form $\text{Av}(\sigma)$ of all the permutations that do not contain a fixed pattern σ .

Note that for the class $\text{Av}(21)$, consisting of all the increasing permutations, $\text{Av}(21)$ -PATTERN PPM corresponds to the problem of finding the longest increasing subsequence in the given text, a well-known polynomially solvable problem [17]. Another polynomially solvable case is $\text{Av}(132)$ -PATTERN PPM, which follows from more general results of Bose et al. [7].

In contrast, for the class $\text{Av}(321)$ of permutations avoiding a decreasing subsequence of length 3 (or equivalently, the class of permutations formed by merging two increasing sequences), $\text{Av}(321)$ -PATTERN PPM is already NP-complete, as shown by Jelínek and Kynčl [15]. In fact, Jelínek and Kynčl show that $\text{Av}(\sigma)$ -PATTERN PPM is polynomial for $\sigma \in \{1, 12, 21, 132, 231, 312, 213\}$ and NP-complete otherwise.

For the more restricted $\text{Av}(\sigma)$ -PPM problem, a polynomial algorithm for $\sigma = 321$ was found by Guillemot and Vialette [12] (see also Albert et al. [2]), and it follows that $\text{Av}(\sigma)$ -PPM is polynomial for any σ of length at most 3. In contrast, the case $\sigma = 4321$ (and by symmetry also $\sigma = 1234$) is NP-complete [15]. It follows that $\text{Av}(\sigma)$ -PPM is NP-complete whenever σ contains 1234 or 4321 as subpermutation, and in particular, it is NP-complete for any σ of length 10 or more.

In this paper, our main motivation is to close the gap between the polynomial and the NP-complete cases of $\text{Av}(\sigma)$ -PPM. We develop a general type of hardness reduction, applicable to any permutation class that contains a suitable grid-like substructure. We then verify that for most choices of σ large enough, the class $\text{Av}(\sigma)$ contains the required substructure. Specifically, we can prove that $\text{Av}(\sigma)$ -PPM is NP-complete in the following cases:

- Any σ of size at least 6.
- Any σ of size 5, except the symmetry type of 41352 (i.e., the two symmetric permutations 41352 and 25314).
- Any σ symmetric to one of 4321, 4312 or 4231.

Note that the list above includes the previously known case $\sigma = 4321$. Our hardness reduction, apart from being more general than previous results, has also the advantage of being more efficient: we reduce an instance of 3-SAT of size m to an instance of PPM of

size $O(m \log m)$. This implies, assuming the exponential time hypothesis (ETH), that none of these NP-complete cases of $\text{Av}(\sigma)$ -PPM can be solved in time $2^{o(n/\log n)}$. Previously, this lower bound was not known to hold even for the unconstrained PPM problem.

Grid classes

The sets of permutations of the form $\text{Av}(\sigma)$, i.e., the sets determined by a single forbidden pattern, are the most common type of permutation sets considered; however, such sets are not necessarily the most convenient tools to understand the precise boundary between polynomial and NP-complete cases of PPM. We will instead work with the more general concept of *permutation class*, which is a set \mathcal{C} of permutations with the property that for any $\pi \in \mathcal{C}$, all the subpermutations of π are in \mathcal{C} as well.

A particularly useful family of permutation classes are the so-called grid classes. When dealing with grid classes, it is useful to represent a permutation $\pi = \pi_1\pi_2 \cdots \pi_n$ by its *diagram*, which is the set of points $\{(i, \pi_i) \mid i = 1, \dots, n\}$. A grid class is defined in terms of a *gridding matrix* \mathcal{M} , whose entries are (possibly empty) permutation classes. We say that a permutation π has an \mathcal{M} -*gridding*, if its diagram can be partitioned, by horizontal and vertical cuts, into an array of rectangles, where each rectangle induces in π a subpermutation from the permutation class in the corresponding cell of \mathcal{M} . The permutation class $\text{Grid}(\mathcal{M})$ then consists of all the permutations that have an \mathcal{M} -gridding.

To a gridding matrix \mathcal{M} we associate a *cell graph*, which is the graph whose vertices are the entries in \mathcal{M} that correspond to infinite classes, with two vertices being adjacent if they belong to the same row or column of \mathcal{M} and there is no other infinite entry of \mathcal{M} between them.

In the griddings we consider in this paper, a prominent role is played by four specific classes, forming two symmetry pairs: one pair are the monotone classes $\text{Av}(21)$ and $\text{Av}(12)$, containing all the increasing and all the decreasing permutations, respectively. Note that any infinite class of permutations contains at least one of $\text{Av}(12)$ and $\text{Av}(21)$ as a subclass, by the Erdős–Szekeres theorem [9].

The other relevant pair of classes involves the so-called *Fibonacci class*, denoted $\oplus 21$, and its mirror image $\ominus 12$. The Fibonacci class can be defined as the class of permutations avoiding the three patterns 321, 312 and 231, or equivalently, it is the class of permutations $\pi = \pi_1\pi_2 \cdots \pi_n$ satisfying $|\pi_i - i| \leq 1$ for every i .

Griddings have been previously used, sometimes implicitly, in the analysis of special cases of PPM, where they were applied both in the design of polynomial algorithms [2, 12], and in NP-hardness proofs [15, 16]. In fact, all the known NP-hardness arguments for special cases of \mathcal{C} -PATTERN PPM are based on the existence of suitable grid subclasses of the class \mathcal{C} . In particular, previous results of the authors [16] imply that for any gridding matrix \mathcal{M} that only involves monotone or Fibonacci cells, $\text{Grid}(\mathcal{M})$ -PATTERN PPM is polynomial when the cell graph of \mathcal{M} is a forest, and it is NP-complete otherwise. Of course, if $\text{Grid}(\mathcal{M})$ -PATTERN PPM is polynomial then $\text{Grid}(\mathcal{M})$ -PPM is polynomial as well. However, the results in this paper identify a broad family of examples where $\text{Grid}(\mathcal{M})$ -PPM is polynomial, while $\text{Grid}(\mathcal{M})$ -PATTERN PPM is known to be NP-complete.

Our main hardness result, Theorem 2, can be informally rephrased as a claim that \mathcal{C} -PPM is hard for a class \mathcal{C} whenever \mathcal{C} contains, for each n and a fixed $\varepsilon > 0$, a grid subclass whose cell graph is a path of length n , and at least εn of its cells are Fibonacci classes. A somewhat less technical consequence, Corollary 4, says that $\text{Grid}(\mathcal{M})$ -PPM is NP-hard whenever the cell graph of \mathcal{M} is a cycle with no three vertices in the same row or column and with at least one Fibonacci cell.

Corollary 4 is, in a certain sense, best possible, since our main tractability result, Theorem 10, states that \mathcal{C} -PPM is polynomial whenever \mathcal{C} is *monotone-griddable*, that is, $\mathcal{C} \subseteq \text{Grid}(\mathcal{M})$, where \mathcal{M} contains only monotone (or empty) cells. Moreover, by a result of Huczynska and Vatter [13], every class \mathcal{C} that does not contain $\oplus 21$ or $\ominus 12$ is monotone griddable. Taken together, these results show that $\text{Grid}(\mathcal{M})$ -PPM is polynomial whenever no cell of \mathcal{M} contains $\oplus 21$ or $\ominus 12$ as a subclass.

2 Preliminaries

A *permutation of length n* is a sequence π_1, \dots, π_n in which each element of the set $[n] = \{1, 2, \dots, n\}$ appears exactly once. When writing out short permutations explicitly, we shall omit all punctuation and write, e.g., 15342 for the permutation 1, 5, 3, 4, 2. The *permutation diagram* of π is the set of points $S_\pi = \{(i, \pi_i) \mid i \in [n]\}$ in the plane. Observe that no two points from S_π share the same x - or y -coordinate. We say that such a set is in *general position*. Note that we blur the distinction between permutations and their permutation diagrams, e.g., we shall refer to “the point of π ”.

For a point p in the plane, we let $p.x$ denote its horizontal coordinate and $p.y$ its vertical coordinate. Two finite sets $S, R \subseteq \mathbb{R}^2$ in general position are *order-isomorphic*, or just *isomorphic* for short, if there is a bijection $f: S \rightarrow R$ such that for any pair of points $p \neq q$ of R we have $f(p).x < f(q).x$ if and only if $p.x < q.x$, and $f(p).y < f(q).y$ if and only if $p.y < q.y$; in such case, the function f is the *isomorphism* from S to R . The *reduction* of a finite set $S \subseteq \mathbb{R}^2$ in general position is the unique permutation π such that S is isomorphic to S_π .

A permutation τ *contains* a permutation π , denoted by $\pi \preceq \tau$, if there is a subset $P \subseteq S_\tau$ that is isomorphic to S_π . Such a subset is then called *an occurrence* of π in τ , and the isomorphism from S to P is an *embedding* of π into τ . If τ does not contain π , we say that τ *avoids* π .

A *permutation class* is any down-set \mathcal{C} of permutations, i.e., a set \mathcal{C} such that if $\pi \in \mathcal{C}$ and $\sigma \preceq \pi$ then also $\sigma \in \mathcal{C}$. For a permutation σ , we let $\text{Av}(\sigma)$ denote the class of all σ -avoiding permutations. We shall throughout use the symbols \boxplus and \boxminus as short-hands for the class of increasing permutations $\text{Av}(21)$ and the class of decreasing permutations $\text{Av}(12)$.

Observe that for every permutation π of length at most m , the permutation diagram S_π is a subset of the set $\{p \mid \frac{1}{2} < p.x < m + \frac{1}{2} \wedge \frac{1}{2} < p.y < m + \frac{1}{2}\}$, called *m -box*. This fact motivates us to extend the usual permutation symmetries to bijections of the whole m -box. In particular, there are eight symmetries generated by:

reversal which reflects the m -box through its vertical axis, i.e., the image of a point p is the point $(m + 1 - p.x, p.y)$,

complement which reflects the m -box through its horizontal axis, i.e., the image of a point p is the point $(p.x, m + 1 - p.y)$,

inverse which reflects the m -box through its ascending diagonal axis, i.e., the image of a point p is the point $(p.y, p.x)$.

We say that a permutation π is *symmetric* to a permutation σ if π can be transformed into σ by any of the eight symmetries generated by reversal, complement and inverse. The *symmetry type*¹ of a permutation σ is the set of all the permutations symmetric to σ .

¹ We chose the term “symmetry type” over the more customary “symmetry class”, to avoid possible confusion with the notion of permutation class.

The symmetries generated by reversal, complement and inverse can be applied not only to individual permutations but also to their classes. Formally, if Ψ is one of the eight symmetries and \mathcal{C} is a permutation class, then $\Psi(\mathcal{C})$ refers to the set $\{\Psi(\sigma) \mid \sigma \in \mathcal{C}\}$. We may easily see that $\Psi(\mathcal{C})$ is again a permutation class.

Consider a pair of permutations π of length n and σ of length m . The *inflation* of a point p of π by σ is the reduction of the point set

$$S_\pi \setminus \{p\} \cup \left\{ \left(p.x + \frac{q.x}{m+1}, p.y + \frac{q.y}{m+1} \right) \mid q \in S_\sigma \right\}.$$

Informally, we replace the point p with a tiny copy of σ .

The *direct sum* of π and σ , denoted by $\pi \oplus \sigma$, is the result of inflating the “1” in 12 with π and then inflating the “2” with σ . Similarly, the *skew sum* of π and σ , denoted by $\pi \ominus \sigma$, is the result of inflating the “2” in 21 with π and then inflating the “1” with σ . If a permutation τ cannot be obtained as direct sum of two shorter permutations, we say that τ is *sum-indecomposable* and if it cannot be obtained as a skew sum of two shorter permutations, we say that it is *skew-indecomposable*. Moreover, we say that a permutation class \mathcal{C} is *sum-closed* if for any $\pi, \sigma \in \mathcal{C}$ we have $\pi \oplus \sigma \in \mathcal{C}$. We define *skew-closed* analogously.

We define the *sum completion* of a permutation π to be the permutation class

$$\oplus\pi = \{\sigma_1 \oplus \sigma_2 \oplus \cdots \oplus \sigma_k \mid \sigma_i \preceq \pi \text{ for all } i \leq k \in \mathbb{N}\}.$$

Analogously, we define the *skew completion* $\ominus\pi$ of π . The class $\oplus 21$ is known as the *Fibonacci class*.

2.1 Grid classes

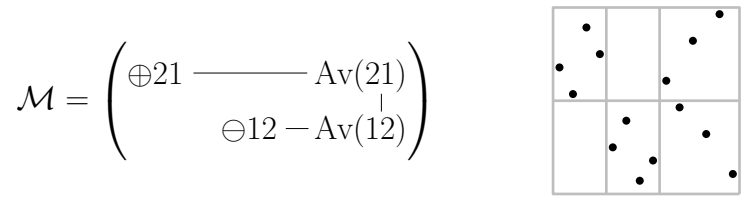
When we deal with matrices, we number their rows from bottom to top to be consistent with the Cartesian coordinates we use for permutation diagrams. For the same reason, we let the column coordinates precede the row coordinates; in particular, a $k \times \ell$ *matrix* is a matrix with k columns and ℓ rows, and for a matrix \mathcal{M} , we let $\mathcal{M}_{i,j}$ denote its entry in column i and row j .

A matrix \mathcal{M} whose entries are permutation classes is called a *gridding matrix*. Moreover, if the entries of \mathcal{M} belong to the set $\{\square, \square, \emptyset\}$ then we say that \mathcal{M} is a *monotone gridding matrix*.

A $k \times \ell$ -*gridding* of a permutation π of length n are two weakly increasing sequences $1 = c_1 \leq \cdots \leq c_{k+1} = n + 1$ and $1 = r_1 \leq \cdots \leq r_{\ell+1} = n + 1$. For each $i \in [k]$ and $j \in [\ell]$, we call the set of points $p \in S_\pi$ such that $c_i \leq p.x < c_{i+1}$ and $r_j \leq p.y < r_{j+1}$ the (i, j) -*cell* of π . An \mathcal{M} -*gridding* of a permutation π is a $k \times \ell$ -gridding such that the reduction of the (i, j) -cell of π belongs to the class $\mathcal{M}_{i,j}$ for every $i \in [k]$ and $j \in [\ell]$. If π has an \mathcal{M} -gridding, then π is said to be \mathcal{M} -*griddable*, and the *grid class* of \mathcal{M} , denoted by $\text{Grid}(\mathcal{M})$, is the class of all \mathcal{M} -griddable permutations.

The *cell graph* of the gridding matrix \mathcal{M} , denoted $G_{\mathcal{M}}$, is the graph whose vertices are pairs (i, j) such that $\mathcal{M}_{i,j}$ is an infinite class, with two vertices being adjacent if they share a row or a column of \mathcal{M} and all the entries between them are finite or empty. See Figure 1. We slightly abuse the notation and use the vertices of $G_{\mathcal{M}}$ for indexing \mathcal{M} , i.e., for a vertex v of $G_{\mathcal{M}}$, we write \mathcal{M}_v to denote the corresponding entry.

A *proper-turning path* in $G_{\mathcal{M}}$ is a path P such that no three vertices of P share the same row or column. Similarly, a *proper-turning cycle* in $G_{\mathcal{M}}$ is a cycle C such that no three vertices of C share the same row or column.



■ **Figure 1** A gridding matrix \mathcal{M} on the left and a permutation equipped with an \mathcal{M} -gridding on the right. Empty entries of \mathcal{M} are omitted and the edges of $G_{\mathcal{M}}$ are displayed inside \mathcal{M} .

Let π be a permutation, and let (c, r) be its $k \times \ell$ -gridding, where $c = (c_1, \dots, c_{k+1})$ and $r = (r_1, \dots, r_{\ell+1})$. A permutation π together with a gridding (c, r) form a *gridded permutation*. When dealing with gridded permutations, it is often convenient to apply symmetry transforms to individual columns or rows of the gridding. Specifically, the *reversal of the i -th column* of π is the operation which generates a new (c, r) -gridded permutation π' by taking the diagram of π , and then reflecting the rectangle $[c_i, c_{i+1} - 1] \times [1, n]$ in the diagram through its vertical axis, producing the diagram of the new permutation π' . Note that π' differs from π by having all the entries at positions $c_i, c_i + 1, \dots, c_{i+1} - 1$ in reverse order. If $c_{i+1} \leq c_i + 1$, then $\pi' = \pi$.

Similarly, the *complementation of the j -th row* of the (c, r) -gridded permutation π is obtained by taking the rectangle $[1, n] \times [r_j, r_{j+1} - 1]$ and turning it upside down, obtaining a permutation diagram of a new permutation.

Column reversals and row complementations can also be applied to gridding matrices: a reversal of a column i in a gridding matrix \mathcal{M} simply replaces all the classes appearing in the entries of the i -th column by their reverses; a row complementation is defined analogously.

We often need to perform several column reversals and row complementations at once. To describe such operations succinctly, we introduce the concept of $k \times \ell$ -orientation. A $k \times \ell$ -orientation is a pair of functions $\mathcal{F} = (f_c, f_r)$ with $f_c: [k] \rightarrow \{-1, 1\}$ and $f_r: [\ell] \rightarrow \{-1, 1\}$. To *apply* the orientation \mathcal{F} to a $k \times \ell$ -gridded permutation π means to create a new permutation $\mathcal{F}(\pi)$ by reversing in π each column i for which $f_c(i) = -1$ and complementing each row j for which $f_r(j) = -1$. Note that the order in which we perform the reversals and complementations does not affect the final outcome. Note also that \mathcal{F} is an involution, that is, $\mathcal{F}(\mathcal{F}(\pi)) = \pi$ for any $k \times \ell$ -gridded permutation π .

We may again also apply \mathcal{F} to a gridding matrix \mathcal{M} . By performing, in some order, the row reversals and column complementations prescribed by \mathcal{F} on the matrix \mathcal{M} , we obtain a new gridding matrix $\mathcal{F}(\mathcal{M})$. For instance, taking the gridding matrix $\begin{pmatrix} \boxtimes & \boxtimes \\ \boxtimes & \boxtimes \end{pmatrix}$ and applying reversal to its first column yields the gridding matrix $\begin{pmatrix} \boxtimes & \boxtimes \\ \boxtimes & \boxtimes \end{pmatrix}$. Observe that if (c, r) is an \mathcal{M} -gridding of a permutation π , then the same gridding (c, r) is also an $\mathcal{F}(\mathcal{M})$ -gridding of the permutation $\mathcal{F}(\pi)$.

Let \mathcal{M} be a monotone gridding matrix. An orientation \mathcal{F} of \mathcal{M} is *consistent* if all the nonempty entries of $\mathcal{F}(\mathcal{M})$ are equal to \boxtimes . For instance, the matrix $\begin{pmatrix} \boxtimes & \boxtimes \\ \boxtimes & \boxtimes \end{pmatrix}$ has a consistent orientation acting by reversing the first column and complementing the first row, while the matrix $\begin{pmatrix} \boxtimes & \boxtimes \\ \boxtimes & \boxtimes \end{pmatrix}$ has no consistent orientation. We remark that Vatter and Waton [18] have shown that any monotone gridding matrix whose cell graph is acyclic has a consistent orientation.

A vital role in our arguments is played by the concept of monotone griddability. We say that a class \mathcal{C} is *monotone-griddable* if there exists a monotone gridding matrix \mathcal{M} such that \mathcal{C} is contained in $\text{Grid}(\mathcal{M})$. Huczynska and Vatter [13] provided a neat and useful characterization of monotone-griddable classes.

► **Theorem 1** (Huczynska and Vatter [13]). *A permutation class \mathcal{C} is monotone-griddable if and only if it contains neither the Fibonacci class $\oplus 21$ nor its symmetry $\ominus 12$.*

Finally, a monotone grid class $\text{Grid}(\mathcal{C} \mathcal{D})$ where both \mathcal{C} and \mathcal{D} are non-empty is called a *horizontal monotone juxtaposition*. Analogously, a *vertical monotone juxtaposition* is a monotone grid class $\text{Grid}(\frac{\mathcal{C}}{\mathcal{D}})$ with both \mathcal{C} and \mathcal{D} non-empty. A *monotone juxtaposition* is simply a class that is either a horizontal or a vertical monotone juxtaposition.

2.2 Pattern matching complexity

In this paper, we deal with the complexity of the decision problem known as \mathcal{C} -PPM. For a permutation class \mathcal{C} , the input of \mathcal{C} -PPM is a pair of permutations (π, τ) with both π and τ belonging to \mathcal{C} . An instance of \mathcal{C} -PPM is then accepted if τ contains π , and rejected if τ avoids π . In the context of pattern-matching, π is referred to as the *pattern*, while τ is the *text*.

Note that an algorithm for \mathcal{C} -PPM does not need to verify that the two input permutations belong to the class \mathcal{C} , and the algorithm may answer arbitrarily on inputs that fail to fulfill this constraint. Decision problems that place this sort of validity restrictions on their inputs are commonly known as *promise problems*.

Our NP-hardness results for \mathcal{C} -PPM are based on a general reduction scheme from the classical 3-SAT problem. Given that \mathcal{C} -PPM is a promise problem, the reduction must map instances of 3-SAT to valid instances of \mathcal{C} -PPM, i.e., the instances where both π and τ belong to \mathcal{C} .

On top of NP-hardness arguments, we also provide time-complexity lower bounds for the hard cases of \mathcal{C} -PPM. These lower bounds are conditioned on the *exponential-time hypothesis* (ETH), a classical hardness assumption which states that there is a constant $\varepsilon > 0$ such that 3-SAT cannot be solved in time $O(2^{\varepsilon n})$, where n is the number of variables of the 3-SAT instance. In particular, ETH implies that 3-SAT cannot be solved in time $2^{o(n)}$.

Given an instance (π, τ) of \mathcal{C} -PPM, we always use n to denote the length of the text τ . We also freely assume that π has length at most n since otherwise the instance can be straightforwardly rejected. Following established practice, we express our complexity bounds for \mathcal{C} -PPM in terms of n . Note that inputs of \mathcal{C} -PPM of size n actually require $\Theta(n \log n)$ bits to encode.

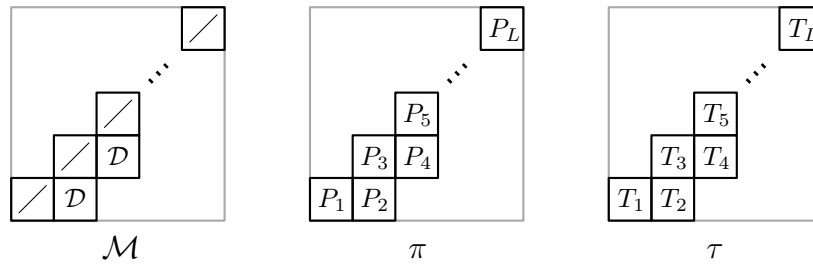
3 Hardness of PPM

In this section, we present the main technical hardness result and then derive its several corollaries. However, we first need to introduce one more definition.

We say that a permutation class \mathcal{C} has the *\mathcal{D} -rich path property* for a class \mathcal{D} if there is a positive constant ε such that for every k , the class \mathcal{C} contains a grid subclass whose cell graph is a proper-turning path of length k with at least $\varepsilon \cdot k$ entries equal to \mathcal{D} . Moreover, we say that \mathcal{C} has the *computable \mathcal{D} -rich path property*, if \mathcal{C} has the \mathcal{D} -rich path property and there is an algorithm that, for a given k , outputs a witnessing proper-turning path of length k with at least $\varepsilon \cdot k$ copies of \mathcal{D} in time polynomial in k .

► **Theorem 2.** *Let \mathcal{C} be a permutation class with the computable \mathcal{D} -rich path property for a non-monotone-griddable class \mathcal{D} . Then \mathcal{C} -PPM is NP-complete, and unless ETH fails, there can be no algorithm that solves \mathcal{C} -PPM*

- *in time $2^{o(n/\log n)}$ if \mathcal{D} moreover contains any monotone juxtaposition,*
- *in time $2^{o(\sqrt{n})}$ otherwise.*



■ **Figure 2** The gridding matrix \mathcal{M} , the gridded permutation π (the pattern) and the gridded permutation τ (the text), used in the simplified overview of the proof of Theorem 2.

We remark, without going into detail, that the two lower bounds we obtained under ETH are close to optimal. It is clear that the bound of $2^{o(n/\log n)}$ matches, up to the $\log n$ term in the exponent, the trivial $2^{O(n)}$ brute-force algorithm for PPM. Moreover, the lower bound of $2^{o(\sqrt{n})}$ for \mathcal{C} -PPM also cannot be substantially improved without adding assumptions about the class \mathcal{C} . Consider for instance the class $\mathcal{C} = \left(\begin{array}{c|c} \square & \square \\ \oplus_{21} & \square \end{array} \right)$. As we shall see in Proposition 3, this class has the computable \oplus_{21} -rich path property, and therefore the $2^{o(\sqrt{n})}$ conditional lower bound applies to it. However, by using the technique of Ahal and Rabinovich [1], which is based on the concept of treewidth of permutations, we can solve \mathcal{C} -PPM (even \mathcal{C} -PATTERN PPM) in time $n^{O(\sqrt{n})}$. This is because we can show that a permutation $\pi \in \mathcal{C}$ of size n has treewidth at most $O(\sqrt{n})$. We omit the details of the argument here.

3.1 Overview of the proof of Theorem 2

The proof of Theorem 2 is based on a reduction from the well-known 3-SAT problem. The individual steps of the construction are rather technical and in view of the space constraints, we only present here a high-level overview of the construction, while some of the more technical aspects are described in the appendix.

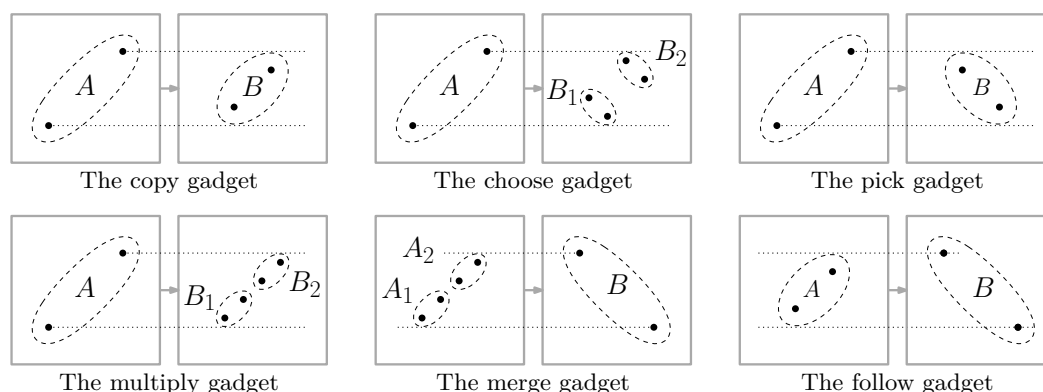
Suppose that \mathcal{C} is a class with the computable \mathcal{D} -rich path property, where \mathcal{D} is not monotone griddable. This means that \mathcal{D} contains the Fibonacci class \oplus_{21} or its reversal \ominus_{12} as subclass. Suppose then, without loss of generality, that \mathcal{D} contains \oplus_{21} .

To reduce 3-SAT to \mathcal{C} -PPM, consider a 3-SAT formula Φ , with n variables x_1, \dots, x_n and m clauses. We may assume that each clause of Φ has exactly 3 literals.

Let $L = L(m, n)$ be an integer whose value will be specified later. By the \mathcal{D} -rich path property, \mathcal{C} contains a grid subclass $\text{Grid}(\mathcal{M})$ where the cell graph of \mathcal{M} is a path of length L , in which a constant fraction of cells is equal to \mathcal{D} .

To simplify our notation in this high-level overview, we will assume that the cell graph of \mathcal{M} corresponds to an increasing staircase. More precisely, the cells of \mathcal{M} representing infinite classes can be arranged into a sequence C_1, C_2, \dots, C_L , where C_1 is the bottom-left cell $\mathcal{M}_{1,1}$ of \mathcal{M} , each odd-numbered cell C_{2i-1} corresponds to the diagonal cell $\mathcal{M}_{i,i}$, and each even numbered cell $C_{2i,2i}$ corresponds to $\mathcal{M}_{i+1,i}$. All the remaining cells of \mathcal{M} are empty. To simplify the exposition even further, we will assume that each odd-numbered cell of the path is equal to \square and each even-numbered cell is equal to \mathcal{D} . See Figure 2.

With the gridding matrix \mathcal{M} specified above, we will construct two \mathcal{M} -gridded permutations, the pattern π and the text τ , such that π can be embedded into τ if and only if the formula Φ is satisfiable. We will describe π and τ geometrically, as permutation diagrams, which are partitioned into blocks by the \mathcal{M} -gridding. We let P_i denote the part of π corresponding to the cell C_i of \mathcal{M} , and similarly we let T_i be the part of τ corresponding to C_i .



■ **Figure 3** The constructions of simple gadgets. The tile Q_i is always on the left and the tile Q_{i+1} is on the right. The dotted lines show the relative vertical order of points.

To get an intuitive understanding of the reduction, it is convenient to first restrict our attention to *grid-preserving embeddings* of π into τ , that is, to embeddings which map the elements of P_i to elements of T_i for each i .

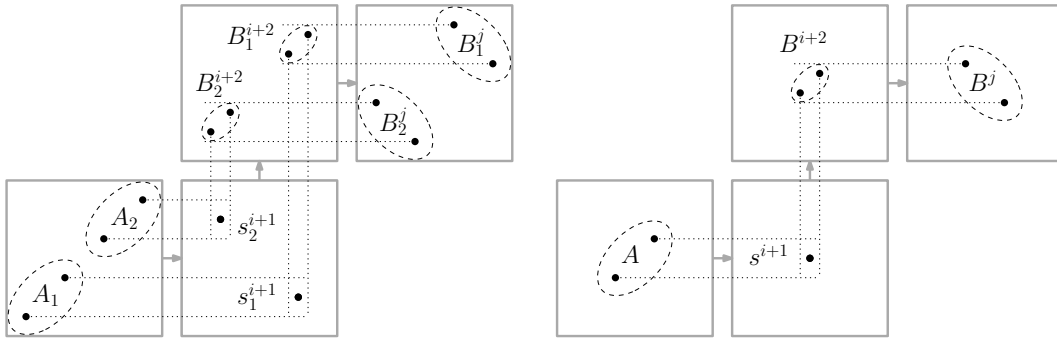
The basic building blocks in the description of π and τ are the *atomic pairs*, which are specific pairs of points appearing inside a single block P_i or T_i . It is a feature of the construction that in any grid preserving embedding of π into τ , an atomic pair inside a pattern block P_i is mapped to an atomic pair inside the corresponding text block T_i . Moreover, each atomic pair in π or τ is associated with one of the variables x_1, \dots, x_n of Φ , and any grid-preserving embedding will maintain the association, that is, atomic pairs associated to a variable x_j inside π will map to atomic pairs associated to x_j in τ .

To describe π and τ , we need to specify the relative positions of the atomic pairs in two adjacent blocks P_i and P_{i+1} (or T_i and T_{i+1}). These relative positions are given by several typical configurations, which we call *gadgets*. Several examples of gadgets are depicted in Figure 3. In the figure, the pairs of points enclosed by an ellipse are atomic pairs. The choose, multiply and merge gadgets are used in the construction of τ , while the pick and follow gadgets are used in π . The copy gadget will be used in both. We also need more complicated gadgets, namely the *flip gadgets* of Figure 4, which span more than two consecutive blocks. In all cases, the atomic pairs participating in a single gadget are all associated to the same variable of Φ .

The sequence of pattern blocks P_1, P_2, \dots, P_L , as well as their corresponding text blocks T_1, \dots, T_L , is divided into several contiguous parts, which we call *phases*. We now describe the individual phases in the order in which they appear.

The initial phase and the assignment phase. The initial phase involves a single pattern block P_1 and the corresponding text block T_1 . Both P_1 and T_1 consist of an increasing sequence of $2n$ points, divided into n consecutive atomic pairs $X_1^1, X_2^1, \dots, X_n^1 \subseteq P_1$ and $Y_1^1, Y_2^1, \dots, Y_n^1 \subseteq T_1$, numbered in increasing order. The pairs X_j^1 and Y_j^1 are both associated to the variable x_j . Clearly any embedding of P_1 into T_1 will map the pair X_j^1 to the pair Y_j^1 , for each $j \in [n]$.

The initial phase is followed by the assignment phase, which also involves only one pattern block P_2 and the corresponding text block T_2 . P_2 will consist of an increasing sequence of n atomic pairs $X_1^2, X_2^2, \dots, X_n^2$, where each X_j^2 is a decreasing pair, i.e., a copy of 21 . Moreover, $X_j^1 \cup X_j^2$ forms the pick gadget, so the first two pattern blocks can be viewed as a sequence of n pick gadgets stacked on top of each other.



■ **Figure 4** A flip text gadget on the left and a flip pattern gadget on the right. The first tile pictured is Q_i and the last tile is Q_j where $j = i + 3$. As before, the dotted lines show the relative order of points.

The block T_2 then consists of $2n$ atomic pairs $\{Y_j^2, Z_j^2; j \in [n]\}$, positioned in such a way that $Y_j^1 \cup Y_j^2 \cup Z_j^2$ is a choose gadget. Thus, $T_1 \cup T_2$ is a sequence of n choose gadgets stacked on top of each other, each associated with one of the variables of Φ .

In a grid-preserving embedding of π into τ , each pick gadget $X_j^1 \cup X_j^2$ must be mapped to the corresponding choose gadget $Y_j^1 \cup Y_j^2 \cup Z_j^2$, with X_j^1 mapped to Y_j^1 , and X_j^2 mapped either to Y_j^2 or to Z_j^2 . There are thus 2^n grid-preserving embeddings of $P_1 \cup P_2$ into $T_1 \cup T_2$, and these embeddings encode in a natural way to the 2^n assignments of truth values to the variables of Φ . Specifically, if X_j^2 is mapped to Y_j^2 , we will say that x_j is false, while if X_j^2 maps to Z_j^2 , we say that x_j is true. The aim is to ensure that an embedding of $P_1 \cup P_2$ into $T_1 \cup T_2$ can be extended to an embedding of π into τ if and only if the assignment encoded by the embedding satisfies Φ .

Each atomic pair that appears in one of the text blocks T_2, T_3, \dots, T_L is not only associated with a variable of Φ , but also with its truth value; that is, there are “true” and “false” atomic pairs associated with each variable x_j . The construction of π and τ ensures that in an embedding of π into τ in which X_j^2 is mapped to Y_j^2 (corresponding to setting x_j to false), all the atomic pairs associated to x_j in the subsequent stages of π will map to false atomic pairs associated to x_j in τ , and conversely, if X_j^2 is mapped to Z_j^2 , then the atomic pairs of π associated to x_j will only map to the true atomic pairs associated to x_j in τ .

The multiplication phase. The purpose of the multiplication phase is to “duplicate” the information encoded in the assignment phase. Without delving into the technical details, we describe the end result of the multiplication phase and its intended behaviour with respect to embeddings. Let d_j be the number of occurrences (positive or negative) of the variable x_j in Φ . Note that $d_1 + d_2 + \dots + d_n = 3m$, since Φ has m clauses, each of them with three literals. Let P_k and T_k be the final pattern block and text block of the multiplication phase. Then P_k is an increasing sequence of $3m$ increasing atomic pairs, among which there are d_j atomic pairs associated to x_j . Moreover, the pairs are ordered in such a way that the d_1 pairs associated to x_1 are at the bottom, followed by the d_2 pairs associated to x_2 and so on. The structure of T_k is similar to P_k , except that T_k has $6m$ atomic pairs. In fact, we may obtain T_k from P_k by replacing each atomic pair $X_i^k \subseteq P_k$ associated to a variable x_j by two adjacent atomic pairs Y_i^k, Z_i^k , associated to the same variable, where Y_i^k is false and Z_i^k is true.

It is useful to identify each pair $X_i^k \subseteq P_k$ as well as the corresponding two pairs $Y_i^k, Z_i^k \subseteq T_k$ with a specific occurrence of x_j in Φ . Thus, each literal in Φ is represented by one atomic pair in P_k and two adjacent atomic pairs of opposite truth values in T_k .

The blocks P_3, \dots, P_k and T_3, \dots, T_k are constructed in such a way that any embedding of π into τ that encodes an assignment in which x_j is false has the property that all the atomic pairs in P_k associated to x_j are mapped to the false atomic pairs of T_k associated to x_j , and similarly, when x_j is encoded as true in the assignment phase, the pairs of P_k associated to x_j are only mapped to the true atomic pairs of T_k . Thus, the mapping of any atomic pair of P_k encodes the information on the truth assignment of the associated variable.

The multiplication phase is implemented by a combination of multiply gadgets and flip text gadgets in τ , and copy gadgets and flip pattern gadgets in π . It requires no more than $O(\log m)$ blocks in π and τ , i.e., $k = O(\log m)$.

The sorting phase. The purpose of the sorting phase is to rearrange the relative positions of the atomic pairs. While at the end of the multiplication phase, the pairs representing occurrences of the same variable appear consecutively, after the sorting phase, the pairs representing literals belonging to the same clause will appear consecutively. More precisely, letting P_ℓ and T_ℓ denote the last pattern block and the last text block of the sorting phase, P_ℓ has the same number of atomic pairs associated to a given variable x_j as P_k , and similarly for T_ℓ and T_k . If K_1, \dots, K_m are the clauses of Φ , then for each clause K_j , P_ℓ contains three consecutive atomic pairs corresponding to the three literals in K_j , and T_ℓ contains the corresponding six atomic pairs, again appearing consecutively. Similarly as in P_k and T_k , each atomic pair in P_ℓ must map to an atomic pair in T_ℓ representing the same literal and having the correct truth value encoded in the assignment phase.

To prove Theorem 2, we need two different ways to implement the sorting phase, depending on whether the class \mathcal{D} contains a monotone juxtaposition or not. The first construction, which we call *sorting by gadgets*, does not put any extra assumptions on \mathcal{D} . However, it may require up to $\Theta(m)$ blocks to perform the sorting, that is $\ell = \Theta(m)$.

The other implementation of the sorting phase, which we call *sorting by juxtapositions* is only applicable when \mathcal{D} contains a monotone juxtaposition, and it can be performed with only $O(\log m)$ blocks. The difference between the lengths of the two versions of sorting is the reason for the two different lower bounds in Theorem 2.

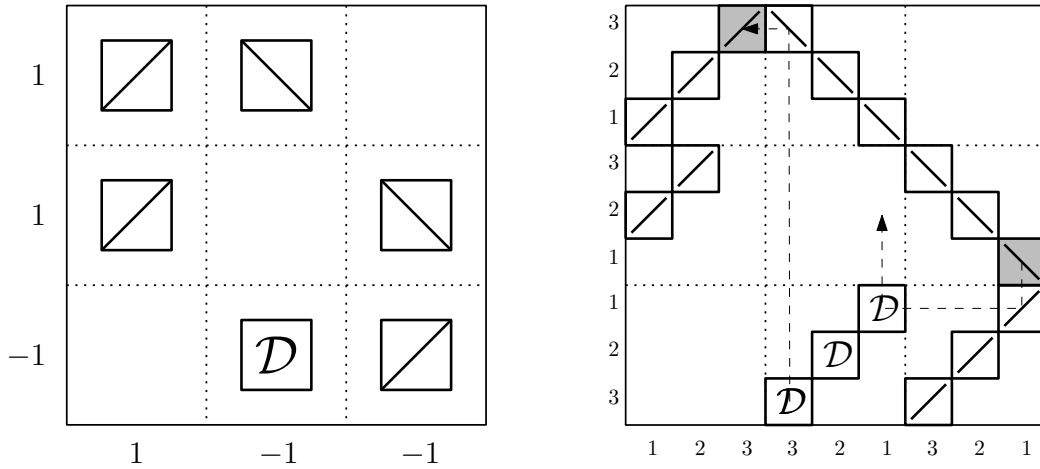
The evaluation phase. The final phase of the construction is the evaluation phase. The purpose of this phase is to ensure that for any embedding of π into τ , the truth assignment encoded by the embedding satisfies all the clauses of Φ . For each clause K_j , we attach suitable gadgets to the atomic pairs in P_ℓ and T_ℓ representing the literals of K_j . Using the fact that the atomic pairs representing the literals of a given clause are consecutive in P_j and T_j , this can be done for all the clauses simultaneously, with only $O(1)$ blocks in π and τ . This completes an overview of the hardness reduction proving Theorem 2.

When the reduction is performed with sorting by gadgets, it produces permutations π and τ of size $O(m^2)$, since we have $L = O(m)$ blocks and each block has size $O(m)$. When sorting is done by juxtapositions, the number of blocks drops to $L = O(\log m)$, hence π and τ have size $O(m \log m)$. ETH implies that 3-SAT with n variables and m clauses cannot be solved in time $2^{o(m+n)}$ [14]. From this, the lower bounds from Theorem 2 follow.

Further details of the reduction, as well as the correctness proof, are presented in the appendix A.

3.2 Consequences

In the rest of this section, we focus on presenting examples of classes that satisfy the technical “rich path” property, which is the backbone of all our hardness arguments.



■ **Figure 5** Illustration of the proof of Proposition 3. Left: a gridding matrix \mathcal{M} whose cell graph is a cycle with a single entry equal to \mathcal{D} . The numbers along the bottom and the left edge form an orientation that maps each entry to a sum-closed class. Right: a gridding matrix whose grid class is contained in $\text{Grid}(\mathcal{M})$ and whose cell graph is a path. The endvertices of the path are highlighted in gray, and the first and last few steps of the path are shown as dashed lines. The numbers along the edges are the labels forming the characteristic of each entry.

► **Proposition 3.** *Let \mathcal{D} be a non-monotone-griddable class that is sum-closed or skew-closed. If \mathcal{M} is a gridding matrix whose cell graph $G_{\mathcal{M}}$ contains a proper-turning cycle with at least one entry equal to \mathcal{D} , then $\text{Grid}(\mathcal{M})$ has the computable \mathcal{D} -rich path property.*

Proof. We note that the proof closely follows a proof of a similar claim for monotone grid classes by Jelínek et al. [16, Lemma 3.5].

We may assume, without loss of generality, that the cell graph of \mathcal{M} consists of a single cycle, that it contains a unique entry equal to \mathcal{D} , and that all the remaining nonempty entries are equal to \boxplus or to \boxminus . This is because each infinite permutation class contains either \boxplus or \boxminus as a subclass, and replacing an entry of \mathcal{M} by its infinite subclass can only change $\text{Grid}(\mathcal{M})$ into its subclass. If we can establish the \mathcal{D} -rich path property for the subclass, then it also holds for the class $\text{Grid}(\mathcal{M})$ itself.

We may also assume that \mathcal{D} is sum-closed, since the skew-closed case is symmetric. In particular, \mathcal{D} contains $\oplus 21$ as a subclass.

Let L be a given integer. We show how to obtain a grid subclass of $\text{Grid}(\mathcal{M})$ whose cell graph is a proper-turning path of length at least L that contains a constant fraction of \mathcal{D} -entries. Refer to Figure 5. Suppose \mathcal{N} is a $k \times \ell$ gridding matrix whose every entry is either sum-closed or skew-closed. The refinement $\mathcal{N}^{\times q}$ of \mathcal{N} is the $qk \times q\ell$ matrix obtained from \mathcal{N} by replacing the entry $\mathcal{N}_{i,j}$ with

- a $q \times q$ diagonal matrix with all the non-empty entries equal to $\mathcal{N}_{i,j}$ if $\mathcal{N}_{i,j}$ is sum-closed,
- a $q \times q$ anti-diagonal matrix with all the non-empty entries equal to $\mathcal{N}_{i,j}$ if $\mathcal{N}_{i,j}$ is skew-closed.

It is easy to see that $\text{Grid}(\mathcal{N}^{\times q})$ is a subclass of $\text{Grid}(\mathcal{N})$. We call the submatrix of $\mathcal{N}^{\times q}$ formed by the entries $\mathcal{N}_{a,b}^{\times q}$ for $q \cdot i < a \leq (q + 1) \cdot i$ and $q \cdot j < b \leq (q + 1) \cdot j$ the (i, j) -block of $\mathcal{N}^{\times q}$.

Importantly, it follows from the work of Albert et al. [3, Proposition 4.1] that for every monotone gridding matrix \mathcal{N} , there exists a consistent orientation of the refinement $\mathcal{N}^{\times 2}$. Translating it to our setting, we can assume that there is a $k \times \ell$ orientation \mathcal{F} such that the image of $\mathcal{M}_{i,j}$ under \mathcal{F} is sum-closed for every $i \in [k]$ and $j \in [\ell]$. If that is not the case for \mathcal{M} , we simply start with $\mathcal{M}^{\times 2}$ instead.

Given such an orientation $\mathcal{F} = (f_c, f_r)$, we label the rows and columns of the refinement $\mathcal{M}^{\times L}$ using the set $[L]$. The L -tuple of columns created from the i -th column of \mathcal{M} is labeled in the increasing order from left to right if $f_c(i)$ is positive and right to left otherwise. Similarly, the L -tuple of rows created from the j -th row of \mathcal{M} is labeled in the increasing order from bottom to top if $f_r(j)$ is positive and top to bottom otherwise. The *characteristic of an entry* in $\mathcal{M}^{\times L}$ is the pair of labels given to its column and row. Observe that each non-empty entry in $\mathcal{M}^{\times L}$ has a characteristic of the form (s, s) for some $s \in [L]$ by the choice of orientation. Therefore, $G_{\mathcal{M}^{\times L}}$ consists exactly of L connected components, each corresponding to a copy of \mathcal{M} .

We pick an arbitrary non-empty monotone entry $\mathcal{M}_{i,j}$ of \mathcal{M} and obtain a matrix \mathcal{M}_L by replacing the (i, j) -block in $\mathcal{M}^{\times q}$ with the $q \times q$ matrix whose only non-empty entries are the ones with characteristic $(s, s+1)$ for all $s \in [L-1]$ and they are all equal to $\mathcal{M}_{i,j}$. $\text{Grid}(\mathcal{M}_L)$ is a subclass of $\text{Grid}(\mathcal{M})$ since the modified (i, j) -block corresponds to shifting the original (anti-)diagonal matrix by one row either up or down, depending on the orientation of the j -th row of \mathcal{M} .

Observe that we connected all the L copies of \mathcal{M} into a single long path. Moreover, the path contains $L-1$ entries in the (i, j) -block and L entries in every other non-empty block. Therefore, a constant fraction of its entries belong to the (a, b) -block such that $\mathcal{M}_{a,b} = \mathcal{D}$ and thus are equal to \mathcal{D} . It is easy to see that the described procedure is constructive and can easily be implemented to run in polynomial time. Therefore, $\text{Grid}(\mathcal{M})$ indeed has the computable \mathcal{D} -rich path property. \blacktriangleleft

Combining Proposition 3 with Theorem 2, we get the following corollary. Note that in the corollary, if \mathcal{D} fails to be sum-closed or skew-closed, we may simply replace it with $\oplus 21$ or $\ominus 12$, since at least one of these two classes is its subclass by Theorem 1.

► Corollary 4. *Let \mathcal{D} be a non-monotone-griddable class. If \mathcal{M} is a gridding matrix whose cell graph contains a proper-turning cycle with one entry equal to \mathcal{D} , then $\text{Grid}(\mathcal{M})$ -PPM is NP-complete. Moreover, unless ETH fails, there can be no algorithm for $\text{Grid}(\mathcal{M})$ -PPM running*

- *in time $2^{o(n/\log n)}$ if \mathcal{D} additionally contains any monotone juxtaposition and is either sum-closed or skew-closed,*
- *in time $2^{o(\sqrt{n})}$ otherwise.*

Three symmetry types of patterns of length 4 can be tackled with a special type of grid classes. The k -step increasing $(\mathcal{C}, \mathcal{D})$ -staircase, denoted by $\text{St}_k(\mathcal{C}, \mathcal{D})$ is a grid class $\text{Grid}(\mathcal{M})$ of a $k \times (k+1)$ gridding matrix \mathcal{M} such that the only non-empty entries in \mathcal{M} are $\mathcal{M}_{i,i} = \mathcal{C}$ and $\mathcal{M}_{i,i+1} = \mathcal{D}$ for every $i \in [k]$. In other words, the entries on the main diagonal are equal to \mathcal{C} and the entries of the adjacent lower diagonal are equal to \mathcal{D} . The increasing $(\mathcal{C}, \mathcal{D})$ -staircase, denoted by $\text{St}(\mathcal{C}, \mathcal{D})$, is the union of $\text{St}_k(\mathcal{C}, \mathcal{D})$ over all $k \in \mathbb{N}$.

Observe that if \mathcal{C} and \mathcal{D} are two infinite classes and one of them contains $\oplus 21$ or $\ominus 12$ then Theorem 2 applies and $\text{St}(\mathcal{C}, \mathcal{D})$ -PPM is NP-complete. Furthermore, if it also contains a monotone juxtaposition as a subclass, then the almost linear lower bound under ETH follows. We proceed to show that three symmetry types of classes avoiding a pattern of length 4 actually contain such a staircase subclass.

► **Proposition 5.** *For any sum-indecomposable permutation σ , the class $\text{St}(\boxplus, \text{Av}(\sigma))$ is contained in the class $\text{Av}(1 \oplus \sigma)$.*

Proof. Suppose for a contradiction that $\sigma' = 1 \oplus \sigma$ belongs to $\text{St}(\boxplus, \text{Av}(\sigma))$. In particular it belongs to $\text{St}_k(\boxplus, \text{Av}(\sigma))$ for some k and there is a witnessing gridding. If the first element is not mapped to one of the \boxplus -entries on the upper diagonal, then the whole σ' must lie in a single $\text{Av}(\sigma)$ -entry on the lower diagonal, which is clearly not possible. Therefore, the first element must be mapped to one of the \boxplus -entries. Notice that the rest of σ' cannot be mapped to any of the \boxplus -entries as it lies below and to the right of the first element. However, it cannot lie in more than one $\text{Av}(\sigma)$ -entry; otherwise, we could express σ as a direct sum of two shorter permutations. Hence, there must be an occurrence of σ in an $\text{Av}(\sigma)$ -entry which is clearly a contradiction. ◀

A direct consequence of Proposition 5 is that taking σ to be 321, 312 or 231, we see that $\text{St}(\boxplus, \text{Av}(321)) \subseteq \text{Av}(4321)$, $\text{St}(\boxplus, \text{Av}(231)) \subseteq \text{Av}(4231)$ and $\text{St}(\boxplus, \text{Av}(312)) \subseteq \text{Av}(4312)$. Note that the first inclusion is rather trivial and the latter two have been previously observed by Berendsohn [5].

We may easily observe that for any pattern σ of size 3, the class $\text{Av}(\sigma)$ contains the Fibonacci class or its reversal, as well as a monotone juxtaposition. Combining Proposition 5 with Theorem 2 yields the following consequence.

► **Corollary 6.** *For any permutation σ that contains a pattern symmetric to 4321, to 4231, or to 4312, the problem $\text{Av}(\sigma)$ -PPM is NP-complete, and unless ETH fails, it cannot be solved in time $2^{o(n/\log n)}$.*

We verified by computer that there are only five symmetry types of patterns of length 5 that do not contain any of 4321, 4213, 4312 or their symmetries – represented by 14523, 24513, 32154, 42513 and 41352. Of these five, four can be handled by Corollary 4 since they contain a specific type of cyclic grid classes, as we now show.

► **Proposition 7.** *The class $\text{Av}(\sigma)$ contains the class $\text{Grid}(\mathcal{M})$ for the gridding matrix*

$$\mathcal{M} = \begin{pmatrix} \boxplus & \boxplus \\ \text{Av}(\pi) & \boxplus \end{pmatrix} \text{ whenever}$$

- $\pi = 132$ and $\sigma = 14523$, or
- $\pi = 231$ and $\sigma = 24513$, or
- $\pi = 321$ and $\sigma \in \{32154, 42513\}$.

Proof. Suppose that σ and π are one of the listed cases. Observe that $\text{Grid}(\mathcal{M})$ is a subclass of $\text{Av}(\sigma)$ if and only if σ is not in $\text{Grid}(\mathcal{M})$. For contradiction, suppose that the class $\text{Grid}(\mathcal{M})$ contains σ . Therefore, there exists a witnessing \mathcal{M} -gridding $1 = c_1 \leq c_2 \leq c_3 = 6$ and $1 = r_1 \leq r_2 \leq r_3 = 6$ of σ .

Let us consider the four choices of σ separately, starting with $\sigma = 14523$: if $c_2 \leq 3$ and $r_2 \leq 3$, the cell $(2, 2)$ of the gridding contains the pattern 21, if $c_2 \leq 4$ and $r_2 \geq 4$, the cell $(2, 1)$ contains 12, if $c_2 \geq 4$ and $r_2 \leq 4$, the cell $(1, 2)$ contains 12, and if $c_2 \geq 5$ and $r_2 \geq 5$, the cell $(1, 1)$ contains 132. In all cases we get a contradiction with the properties of the \mathcal{M} -gridding. The same argument applies to $\sigma = 14513$, except in the last case we use the pattern 231 instead of 132.

For $\sigma = 32154$, the four cases to consider are $c_2 \leq 4 \wedge r_2 \leq 4$, $c_2 \geq 5 \wedge r_2 \leq 3$, $c_2 \leq 3 \wedge r_2 \geq 5$, and $c_2 \geq 4 \wedge r_2 \geq 4$, in each case getting contradiction in a different cell of the gridding. For $\sigma = 42513$, the analogous argument distinguishes the cases $c_2 \leq 3 \wedge r_2 \leq 3$, $c_2 \geq 4 \wedge r_2 \leq 4$, $c_2 \leq 4 \wedge r_2 \geq 4$, and $c_2 \geq 5 \wedge r_2 \geq 5$. ◀

It is easy to see that every σ of length at least 6 contains a pattern of size 5 which is not symmetric to 41352. Therefore, $\text{Av}(\sigma)$ -PPM is NP-complete for all permutations σ of length at least 4 except for one symmetry type of length 5 and for four out of seven symmetry types of length 4. As $\text{Av}(\sigma)$ -PPM is polynomial-time solvable for any σ of length at most 3, these are, in fact, the only cases left unsolved.

► **Corollary 8.** *If σ is a permutation of length at least 4 that is not in symmetric to any of 3412, 3142, 4213, 4123 or 41352, then $\text{Av}(\sigma)$ -PPM is NP-complete, and unless ETH fails, it cannot be solved in time $2^{o(n/\log n)}$.*

To conclude this section, we remark that the suitable grid subclasses were discovered via computer experiments facilitated by the Permuta library [4].

4 Polynomial-time algorithm

We say that a permutation π is *t-monotone* if there is a partition $\Pi = (S_1, \dots, S_t)$ of S_π such that S_i is a monotone point set for each $i \in [t]$. The partition Π is called a *t-monotone partition*.

Given a *t-monotone* partition $\Pi = (S_1, \dots, S_t)$ of a permutation π and a *t-monotone* partition $\Sigma = (S'_1, \dots, S'_t)$ of τ , an embedding ϕ of π into τ is a (Π, Σ) -embedding if $\phi(S_i) \subseteq S'_i$ for every $i \in [t]$. Guillemot and Marx [11] showed that if we fix a *t-monotone* partitions of both π and τ , the problem of finding a (Π, Σ) -embedding is polynomial-time solvable.

► **Proposition 9** (Guillemot and Marx [11]). *Given a permutation π of length m with a *t-monotone* partition Π and a permutation τ of length n with a *t-monotone* partition Σ , we can decide if there is a (Π, Σ) -embedding of π into τ in time $O(m^2n^2)$.*

We can combine this result with the fact that there is only a bounded number of ways how to grid a permutation, and obtain the following counterpart to Corollary 4.

► **Theorem 10.** *\mathcal{C} -PPM is polynomial-time solvable for any monotone-griddable class \mathcal{C} .*

Proof. Let \mathcal{M} be a $k \times \ell$ monotone gridding matrix such that $\text{Grid}(\mathcal{M})$ contains the class \mathcal{C} . We have to decide whether π is contained in τ for two given permutations π of length m and τ of length n , both belonging to the class \mathcal{C} .

First, we find an \mathcal{M} -gridding of τ . We enumerate all possible $k \times \ell$ gridgings and for each, we test if it is a valid \mathcal{M} -gridding. Observe that there are in total $O(n^{k+\ell-2})$ such gridgings since they are determined by two sequences of values from the set $[n]$, one of length $k - 1$ and the other of length $\ell - 1$. Moreover, it is straightforward to test in time $O(n^2)$ whether a given $k \times \ell$ gridding is in fact an \mathcal{M} -gridding. Note that we are guaranteed to find an \mathcal{M} -gridding as τ belongs to $\mathcal{C} \subseteq \text{Grid}(\mathcal{M})$. We set Σ to be the $(k \cdot \ell)$ -monotone partition of τ into the monotone sequences given by the individual cells of the gridding.

In the second step, we enumerate all possible \mathcal{M} -griddings of π . As with τ , we enumerate all possible $O(m^{k+\ell-2})$ $k \times \ell$ gridgings of π and check for each gridding whether it is actually an \mathcal{M} -gridding in time $O(m^2)$. For each \mathcal{M} -gridding found, we let Π be the $(k \cdot \ell)$ -monotone partition of π given by the gridding, and we apply Proposition 9 to test whether there is a (Π, Σ) -embedding in time $O(m^2n^2)$.

If there is an embedding ϕ of π into τ , there is a $(k \cdot \ell)$ -monotone partition Σ' of π such that ϕ is a (Π, Σ') -embedding. Therefore, the algorithm correctly solves \mathcal{C} -PPM in time $O(n^{k+\ell} + m^{k+\ell}n^2)$ – polynomial in n, m . ◀

Notice that if \mathcal{M} is a gridding matrix whose every entry is monotone griddable, or equivalently no entry contains the Fibonacci class or its reverse as a subclass, then the class $\text{Grid}(\mathcal{M})$ is monotone griddable as well. It follows that for such \mathcal{M} , the $\text{Grid}(\mathcal{M})$ -PPM problem is polynomial-time solvable. We also note that recent results on \mathcal{C} -PATTERN PPM [16] imply that if a gridding matrix \mathcal{M} has an acyclic cell graph, and if every nonempty cell is either monotone or symmetric to a Fibonacci class, then $\text{Grid}(\mathcal{M})$ -PATTERN PPM, and therefore also $\text{Grid}(\mathcal{M})$ -PPM, is polynomial-time solvable as well. These two tractability results contrast with our Corollary 4, which shows that for any gridding matrix \mathcal{M} whose cell graph is a cycle, and whose nonempty cells are all monotone except for one Fibonacci cell, $\text{Grid}(\mathcal{M})$ -PPM is already NP-hard.

5 Open problems

We have presented a hardness reduction which allowed us to show that the $\text{Av}(\sigma)$ -PPM problem is NP-complete for every permutation σ of size at least 6, as well as for most shorter choices of σ . Nevertheless, for several symmetry types of σ , the complexity of $\text{Av}(\sigma)$ -PPM remains open. We collect all the remaining unresolved cases as our first open problem.

► **Open problem 1.** *What is the complexity of $\text{Av}(\sigma)$ -PPM, when σ is a permutation from the set $\{3412, 3142, 4213, 4123, 41352\}$?*

Our hardness results are accompanied by time complexity lower bounds based on the ETH. Specifically, for our NP-hard cases, we show that under ETH, no algorithm may solve \mathcal{C} -PPM in time $2^{o(\sqrt{n})}$. The lower bound can be improved to $2^{o(n/\log n)}$ under additional assumptions about \mathcal{C} . This opens the possibility of a more refined complexity hierarchy within the NP-hard cases of \mathcal{C} -PPM. In particular, we may ask for which \mathcal{C} can \mathcal{C} -PPM be solved in subexponential time.

► **Open problem 2.** *Which cases of \mathcal{C} -PPM can be solved in time $2^{O(n^{1-\varepsilon})}$? Can the general PPM problem be solved in time $2^{o(n)}$?*

References

- 1 Shlomo Ahal and Yuri Rabinovich. On complexity of the subpattern problem. *SIAM J. Discrete Math.*, 22(2):629–649, 2008. doi:10.1137/S0895480104444776.
- 2 Michael Albert, Marie-Louise Lackner, Martin Lackner, and Vincent Vatter. The complexity of pattern matching for 321-avoiding and skew-merged permutations. *Discrete Math. Theor. Comput. Sci.*, 18(2):Paper No. 11, 17, 2016. URL: <https://dmtcs.episciences.org/2607>.
- 3 Michael H. Albert, M. D. Atkinson, Mathilde Bouvel, Nik Ruškuc, and Vincent Vatter. Geometric grid classes of permutations. *Trans. Amer. Math. Soc.*, 365(11):5859–5881, 2013. doi:10.1090/S0002-9947-2013-05804-7.
- 4 Ragnar Pall Ardal, Tomas Ken Magnusson, Émile Nadeau, Bjarni Jens Kristinsson, Bjarki Agust Gudmundsson, Christian Bean, Henning Ulfarsson, Jon Steinn Eliasson, Murray Tannock, Alfur Birkir Bjarnason, Jay Pantone, and Arnar Bjarni Arnarson. *Permuta*, 2021. doi:10.5281/zenodo.4725759.
- 5 Benjamin Aram Berendsohn. Complexity of permutation pattern matching. Master’s thesis, Freie Universität Berlin, Berlin, 2019. URL: https://www.mi.fu-berlin.de/inf/groups/ag-ti/theses/master_finished/berendsohn_benjamin/index.html.
- 6 Benjamin Aram Berendsohn, László Kozma, and Dániel Marx. Finding and counting permutations via CSPs. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, volume 148 of *LIPICs*, pages 1:1–1:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.IPEC.2019.1.

- 7 Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. Pattern matching for permutations. *Inform. Process. Lett.*, 65(5):277–283, 1998. doi:10.1016/S0020-0190(97)00209-3.
- 8 Marie-Louise Bruner and Martin Lackner. A fast algorithm for permutation pattern matching based on alternating runs. *Algorithmica*, 75(1):84–117, 2016. doi:10.1007/s00453-015-0013-y.
- 9 P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935. URL: http://www.numdam.org/item?id=CM_1935__2__463_0.
- 10 Jacob Fox. Stanley–Wilf limits are typically exponential. [arXiv:1310.8378v1](https://arxiv.org/abs/1310.8378v1), 2013.
- 11 Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 82–101. ACM, New York, 2014. doi:10.1137/1.9781611973402.7.
- 12 Sylvain Guillemot and Stéphane Vialette. Pattern matching for 321-avoiding permutations. In *Algorithms and computation*, volume 5878 of *Lecture Notes in Comput. Sci.*, pages 1064–1073. Springer, Berlin, 2009. doi:10.1007/978-3-642-10631-6_107.
- 13 Sophie Huczynska and Vincent Vatter. Grid classes and the Fibonacci dichotomy for restricted permutations. *Electron. J. Combin.*, 13(1):Research Paper 54, 14, 2006. URL: http://www.combinatorics.org/Volume_13/Abstracts/v13i1r54.html.
- 14 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k -SAT. *J. Comput. System Sci.*, 62(2):367–375, 2001. Special issue on the Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999). doi:10.1006/jcss.2000.1727.
- 15 Vít Jelínek and Jan Kynčl. Hardness of permutation pattern matching. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 378–396. SIAM, Philadelphia, PA, 2017. doi:10.1137/1.9781611974782.24.
- 16 Vít Jelínek, Michal Opler, and Jakub Pekárek. A complexity dichotomy for permutation pattern matching on grid classes. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24–28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 52:1–52:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.52.
- 17 C. Schensted. Longest increasing and decreasing subsequences. *Canad. J. Math.*, 13:179–191, 1961. doi:10.4153/CJM-1961-015-3.
- 18 Vincent Vatter and Steve Waton. On partial well-order for monotone grid classes of permutations. *Order*, 28(2):193–199, 2011. doi:10.1007/s11083-010-9165-1.

A The proof of Theorem 2

Our job is to construct a pair of permutations π and τ , both having a gridding corresponding to a \mathcal{D} -rich path, with the property that the embeddings of π into τ will simulate satisfying assignments of a given 3-SAT formula. First, we introduce the concept of \mathcal{F} -assembly which enables us to describe constructions of gridded permutations from a grid class $\text{Grid}(\mathcal{M})$ somewhat independently from the actual shape of \mathcal{M} .

A.1 \mathcal{F} -assembly

A finite subset P of the m -box in general position is called an m -tile and a $k \times \ell$ family of m -tiles is a set $\mathcal{P} = \{P_{i,j} \mid i \in [k], j \in [\ell]\}$ where each $P_{i,j}$ is an m -tile. Let $\mathcal{F} = (f_c, f_r)$ be a $k \times \ell$ orientation and let \mathcal{P} be a family of m -tiles $P_{i,j}$ for $i \in [k], j \in [\ell]$. The \mathcal{F} -assembly of \mathcal{P} is the point set S defined as follows.

We define for every $i \in [k], j \in [\ell]$ the point set $P'_{i,j} = \{p + (i \cdot m, j \cdot m) \mid p \in \Phi_i(\Psi_j(P_{i,j}))\}$ where Φ_i is an identity if $f_c(i) = 1$ and reversal otherwise, while Ψ_j is an identity if $f_r(j) = 1$ and complement otherwise. We set $S = \bigcup P'_{i,j}$ to be the \mathcal{F} -assembly of \mathcal{P} . If S is not in general position, we rotate it clockwise by a tiny angle to a general position without changing the order of any points that originally did not share a common coordinate.

Let \mathcal{M} be a $k \times \ell$ gridding matrix. We say that the *image of $\mathcal{M}_{i,j}$ under \mathcal{F}* is the class $\Phi_i(\Psi_j(\mathcal{M}_{i,j}))$. The *\mathcal{F} -image of \mathcal{M}* , denoted by $\mathcal{F}(\mathcal{M})$, is then the $k \times \ell$ gridding matrix defined as $\mathcal{F}(\mathcal{M})_{i,j} = \Phi_i(\Psi_j(\mathcal{M}_{i,j}))$.

► **Observation 11.** *Let \mathcal{M} be a $k \times \ell$ gridding matrix, let \mathcal{F} be a $k \times \ell$ orientation and \mathcal{P} a $k \times \ell$ family of m -tiles. If for every $i \in [k]$ and $j \in [\ell]$ the reduction of $P_{i,j}$ belongs to the class $\mathcal{F}(\mathcal{M})_{i,j}$ then the reduction of the \mathcal{F} -assembly of \mathcal{P} belongs to $\text{Grid}(\mathcal{M})$.*

Furthermore, if the cell graph $G_{\mathcal{M}}$ of a monotone gridding matrix \mathcal{M} is acyclic, we can always find an orientation \mathcal{F} such that the image $\mathcal{F}(\mathcal{M})_{i,j}$ of every non-empty entry $\mathcal{M}_{i,j}$ is the class \square . We say that such \mathcal{F} is a *consistent orientation* for the gridding matrix \mathcal{M} . The existence of consistent orientations is guaranteed for matrices with acyclic cell graphs.

► **Lemma 12** (Vatter and Waton [18]). *There exists a consistent $k \times \ell$ orientation \mathcal{F} for any monotone $k \times \ell$ gridding matrix \mathcal{M} whose cell graph $G_{\mathcal{M}}$ is acyclic.*

A.2 The reduction

Let Φ be a given 3-CNF formula with n variables x_1, x_2, \dots, x_n and m clauses K_1, K_2, \dots, K_m each containing exactly three literals. Let \mathcal{M} be a $g \times h$ gridding matrix such that $\text{Grid}(\mathcal{M})$ is a subclass of \mathcal{C} , the cell graph $G_{\mathcal{M}}$ is a proper-turning path of sufficient length to be determined later with a constant fraction of its entries is equal to \mathcal{D} .

First, we label the vertices of the path as p_1, p_2, p_3, \dots choosing the direction such that at least half of the \mathcal{D} -entries share a row with their predecessor. By application of Lemma 12, there is a $g \times h$ orientation \mathcal{F} such that the class $\mathcal{F}(\mathcal{M})_{i,j}$ is equal to \square for every monotone entry $\mathcal{M}_{i,j}$ and the class $\mathcal{F}(\mathcal{M})_{i,j}$ contains $\oplus 21$ for every \mathcal{D} -entry $\mathcal{M}_{i,j}$. Our plan is to simultaneously construct two $g \times h$ families of tiles \mathcal{P} and \mathcal{T} and then set π and τ to be the \mathcal{F} -assemblies of \mathcal{P} and \mathcal{T} , respectively. We abuse the notation and for any $g \times h$ family of tiles \mathcal{Q} (in particular for \mathcal{P} and \mathcal{T}), we use Q_i instead of Q_{p_i} to denote the tile corresponding to the i -th cell of the path.

For now, we will only consider restricted embeddings. We say that an embedding of π into τ where π is an \mathcal{F} -assembly of \mathcal{P} and τ is an \mathcal{F} -assembly of \mathcal{T} , is *grid-preserving* if the image of tile $P_{i,j}$ is mapped to the image of $T_{i,j}$ for every i and j . We slightly abuse the notation in the case of grid-preserving embeddings and say that a point q in the tile $P_{i,j}$ is mapped to a point r in the tile $T_{i,j}$ instead of saying that the image of q under the \mathcal{F} -assembly is mapped to the image of r under the \mathcal{F} -assembly. We say that a pair of points r, q in the tile Q_i *sandwiches* a set of points A in the tile Q_{i+1} if for every point $t \in A$ $r.y < t.y < q.y$ in case p_i and p_{i+1} occupy a common row or otherwise, if the same holds for the x -coordinates.

A.2.1 Gadgets

We construct the tiles from gadgets consisting of pairs of points that we call *atomic pairs*. We assume that the tiles are formed as direct sums of the individual gadgets. Consequently, if $A, B \subseteq Q_i$ are point sets of two different gadgets, then either whole A lies to the right and above B or vice versa. We describe the gadgets in the case when p_i and p_{i+1} share a common row and p_i is to the left of p_{i+1} , as the other cases are symmetric. The gadgets are fully described by the relative positions of their points, therefore we refer the reader to Figure 3 for their definitions.

We say that the copy, pick and follow gadgets connect the pair A to the pair B and the multiply gadget multiplies the pair A to B_1 and B_2 . The choose gadget is said to branch the pair A to B_1 and B_2 and the merge gadget merges the pairs A_1 and A_2 into the pair B . We follow with two observations about the behavior of these gadgets.

► **Observation 13.** *Suppose there is a choose gadget branching an atomic pair A^T in the tile T_i to two atomic pairs B_1^T and B_2^T in the tile T_{i+1} , and a pick gadget in \mathcal{P} connecting an atomic pair A^P in the tile P_i to an atomic pair B^P in the tile P_{i+1} . In any grid-preserving embedding of π into τ , if A^P is mapped to A^T then B^P is mapped either to B_1^T or to B_2^T .*

► **Observation 14.** *Suppose there is a merge gadget merging atomic pairs A_1^T and A_2^T in the tile T_i into an atomic pair B^T in the tile T_{i+1} , and a follow gadget connecting an atomic pair A^P in the tile P_i to an atomic pair B^P in the tile P_{i+1} . In any grid-preserving embedding of π into τ , if A^P is mapped to A_α^T for some $\alpha \in \{1, 2\}$ then B^P is mapped to B^T .*

The flip gadget

We proceed to define two gadgets – a flip text gadget and a flip pattern gadget. It is insufficient to consider just two neighboring tiles as we need two \mathcal{D} -entries for the construction. To that end, let i and j be indices such that both p_{i+1} and p_j are \mathcal{D} -entries and there is no other \mathcal{D} -entry between them.

As before, suppose that A_1 and A_2 are two atomic pairs in Q_i . The *flip text gadget* attached to the atomic pairs A_1 and A_2 consists of two points s_1^{i+1}, s_2^{i+1} in the tile Q_{i+1} and two atomic pairs B_1^k and B_2^k in each tile Q_k for every $k \in [i+2, j] = \{i+2, i+3, \dots, j\}$. The points s_1^{i+1}, s_2^{i+1} form an occurrence of 21 and s_α^{i+1} is sandwiched by A_α for each $\alpha \in \{1, 2\}$.

The atomic pairs B_1^k, B_2^k for $k \in [i+2, j-1]$ are set such that B_2^k lies to the left and below of B_1^k , and together, they form an occurrence of 1234. The only difference in the case of atomic pairs B_1^j, B_2^j is that they form an occurrence of 2143. For every $k \in [i+3, j]$ and $\alpha \in \{1, 2\}$, the pair B_α^k sandwiches the pair B_α^{k-1} and moreover, the pair B_α^{i+2} sandwiches the point s_α^{i+1} . We say that the flip text gadget flips the pairs A_1, A_2 in Q_i to the pairs B_2^j, B_1^j in Q_j . See the left part of Figure 4.

We define the *flip pattern gadget* as a set of points isomorphic to the pairs B_1^k for all k together with the point s_1^{i+2} . See the right part of Figure 4.

Observe that flip gadget propagates the mapping properties while switching the order of the pairs. However, it can also be used to test if only one of its initial atomic pairs is used in the embedding. We omit proofs of the following lemmas due to space constraints.

► **Lemma 15.** *Suppose there is a flip pattern gadget connecting an atomic pair \bar{A} in P_i with an atomic pair \bar{B} in P_j . Furthermore, suppose that there is a flip text gadget flipping atomic pairs A_1 and A_2 in T_i to atomic pairs B_2 and B_1 in T_j . In any grid-preserving embedding of π into τ , if \bar{A} is mapped to A_α for some $\alpha \in \{1, 2\}$ then \bar{B} is mapped to B_α .*

► **Lemma 16.** *Suppose that there are two flip pattern gadgets connecting an atomic pair A_α^P in P_i to an atomic pair B_α^P in P_j for $\alpha \in \{1, 2\}$. Suppose that there is a flip text gadget in \mathcal{T} that flips atomic pairs A_1^T and A_2^T in T_i to atomic pairs B_2^T and B_1^T in T_j . There cannot exist a grid-preserving embedding ϕ of π into τ that maps A_α^P to A_α^T for each $\alpha \in \{1, 2\}$.*

Note that all the gadgets except for the copy and multiply ones require a non-monotone entry and thus, we need to somehow bridge the segments of the path consisting only of monotone entries. By *attaching a gadget* to the pair A , we mean connecting to A a chain of copy gadgets leading all the way to its first non-monotone entry and then attaching the desired gadget at the end of this chain.

A.2.2 Constructing the \mathcal{C} -PPM instance

We define the initial tile P_1 to contain atomic pairs X_k^0 for $k \in [n]$ and the initial tile T_1 to contain atomic pairs Y_k^0 for $k \in [n]$ where each X_k^0 and Y_k^0 form an occurrence of 12 and X_i^0 (Y_j^0) lies to the left and below of X_j^0 (Y_i^0) for every $i < j$. Any grid-preserving embedding of π into τ must obviously map X_k^0 to Y_k^0 for every $k \in [n]$. We describe the rest of the construction in four distinct phases.

Assignment phase. In the first phase, we simulate the assignment of truth values to the variables. To that end, we attach to each pair Y_k^0 for $k \in [n]$ a choose gadget that branches Y_k^0 to two atomic pairs $Y_{k,1}^1$ and $Z_{k,1}^1$ and we attach to each pair X_k^0 for $k \in [n]$ a pick gadget that connects X_k^0 to an atomic pair $X_{k,1}^1$. The properties of choose and pick gadgets imply that in any grid-preserving embedding, $X_{k,1}^1$ is either mapped to $Y_{k,1}^1$ or to $Z_{k,1}^1$.

Multiplication phase. Our next goal is to multiply the atomic pairs corresponding to a single variable into as many pairs as there are occurrences of this variable in the clauses. We describe the gadgets dealing with each variable individually.

Fix $k \in [n]$ and let m_k for $k \in [n]$ denote the total number of occurrences of x_k and $\neg x_k$ in Φ . We are going to describe the construction inductively in $\lceil \log m_k \rceil$ steps. Fix $i \geq 1$. We add for each $j \in [2^i]$ three multiply gadgets, one that multiplies the atomic pair $X_{k,j}^i$ to atomic pairs $\tilde{X}_{k,2j-1}^{i+1}$ and $\tilde{X}_{k,2j}^{i+1}$, one that multiplies the pair $Y_{k,j}^i$ to $\tilde{Y}_{k,2j-1}^{i+1}$ and $\tilde{Y}_{k,2j}^{i+1}$, and finally one that multiplies $Z_{k,j}^i$ to $\tilde{Z}_{k,2j-1}^{i+1}$ and $\tilde{Z}_{k,2j}^{i+1}$. Observe that the properties of gadgets imply that for arbitrary $j \in [2^{i+1}]$, $\tilde{X}_{k,j}^{i+1}$ maps either to $\tilde{Y}_{k,j}^{i+1}$ or to $\tilde{Z}_{k,j}^{i+1}$. However in the text, we have the quadruple $\tilde{Y}_{k,2j-1}^{i+1}, \tilde{Y}_{k,2j}^{i+1}, \tilde{Z}_{k,2j-1}^{i+1}, \tilde{Z}_{k,2j}^{i+1}$ in this specific order.

To solve this, we add for each $j \in [2^i]$ a flip text gadget that flips $\tilde{Y}_{k,2j}^{i+1}, \tilde{Z}_{k,2j-1}^{i+1}$ to atomic pairs $Z_{k,2j-1}^{i+1}, Y_{k,2j}^{i+1}$. The properties of flip gadgets (Lemma 15) guarantee that for every $j \in [2^{i+1}]$, the pair $X_{k,j}^{i+1}$ is mapped either to $Y_{k,j}^{i+1}$ or to $Z_{k,j}^{i+1}$. Moreover, the order of atomic pairs in the text now alternates between Y and Z as desired. It follows that we need in total $O(\log m)$ \mathcal{D} -entries for the multiplication phase.

Sorting phase. The multiplication phase ended with atomic pairs $X_{k,j}^i$ in the pattern ordered lexicographically by (k, j) , i.e., bundled in blocks by the variables. The goal of the sorting phase is to rearrange them such that they become bundled by clauses.

First, we remark that it is possible to flip the order of any two neighboring pairs in the pattern using only a $O(1)$ layers of gadgets. Unfortunately, the space constraints make it impossible to include the description here as it is quite involved and technical. Using this approach, we can flip an arbitrary set of neighboring pairs in $O(1)$ layers of gadgets and thus, we can arbitrarily reshuffle the atomic pairs of the pattern in $O(m)$ layers.

On the other hand, we describe how we can do the sorting phase using significantly fewer fewer tiles if the class \mathcal{D} contains a monotone juxtaposition. We shall discuss here only the case when \mathcal{D} contains the juxtaposition $\mathcal{B} = \text{Grid}(\square \square)$ as the other cases can be solved using the same technique. Let p_i be an entry such that $\mathcal{F}(\mathcal{M})_{p_i}$ contains \mathcal{B} and recall that we assumed that p_i shares a common row with p_{i-1} . We construct a tile Q_i from two increasing sets Q_i^1 and Q_i^2 placed next to each other. In particular, we can attach to any atomic pair A in Q_{i-1} a copy gadget connecting A to an atomic pair B and choose arbitrarily whether B lies in Q_i^1 or Q_i^2 .

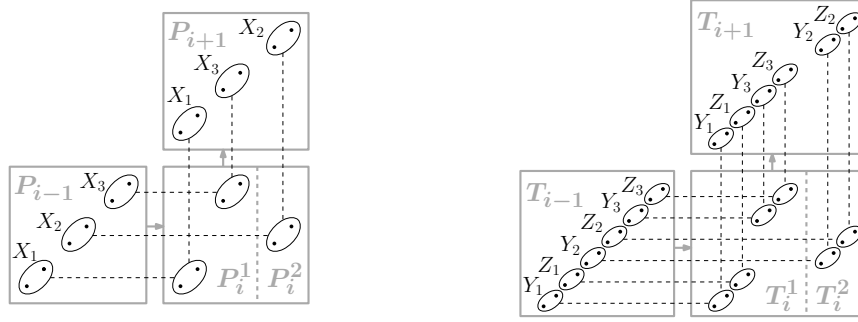


Figure 6 Example of one sorting step using Grid($\square \square$) and a partition $\{1, 2, 3\} = \{1, 3\} \cup \{2\}$.

Let J_1 and J_2 be a partition of the set $[3m]$. We attach a copy gadget ending in Q_i^α to each X_j, Y_j and Z_j with $j \in J_\alpha$ for each $\alpha \in \{1, 2\}$. In this way, we rearranged the atomic pairs in \mathcal{P} such that first we have all pairs X_j such that $j \in J_1$ followed by all pairs X_j for $j \in J_2$. Similarly in \mathcal{T} , we have Y_j, Z_j for $j \in J_1$ followed by Y_j, Z_j for $j \in J_2$. See Figure 6.

Notice that the described operation simulates a stable bucket sort with two buckets. Therefore, we can simulate radix sort and rearrange the atomic pairs into arbitrary order given by σ by iterating this operation $O(\log m)$ times. In this way, the whole sorting phase uses only $O(\log m)$ entries equal to \mathcal{D} .

Evaluation phase. In the evaluation phase, we test whether each clause $K_j = (x_a \vee x_b \vee x_c)$ is satisfied. We consider the case when K_j contains only positive literals as clauses with negative literals can be handled with minor modifications of the argument. Suppose X_a, X_b and X_c are the three neighboring atomic pairs in \mathcal{P} that correspond to the three literals in K_j . In \mathcal{T} , there are six neighboring atomic pairs $Y_a, Z_a, Y_b, Z_b, Y_c, Z_c$ such that in any grid-preserving embedding, the pair X_α is mapped to either Y_α or Z_α for every $\alpha \in \{a, b, c\}$.

We abuse the notation and use the same letters to denote atomic pairs in different tiles so that the gadgets carry the names through. First, we add (i) a choose gadget that branches Z_b to \bar{Z}_b and \tilde{Z}_b , and (ii) pick gadgets to $Y_a, Z_a, Y_b, Y_c, Z_c, X_a, X_b$ and X_c . We continue with adding two layers of flip gadgets, modifying the order of atomic pairs in the text as follows

$$Y_a Z_a \overleftrightarrow{Y_b} \bar{Z}_b \tilde{Z}_b \overleftrightarrow{Y_c} Z_c \rightarrow Y_a \overleftrightarrow{Z_a} \bar{Z}_b Y_b \tilde{Z}_b \overleftrightarrow{Z_c} Y_c \rightarrow Y_a \bar{Z}_b Z_a Y_b Z_c \tilde{Z}_b Y_c.$$

Observe that either X_b is mapped to Y_b and thus K_j is satisfied, or X_b is mapped to one of \bar{Z}_b and \tilde{Z}_b . Subsequently, the order of pairs in the final tile guarantees by Lemma 16 that simultaneously, X_a cannot map to Z_a and X_c to Z_c and thus, K_j must be satisfied.

That concludes the construction of \mathcal{P} and \mathcal{T} . Observe that each tile in both \mathcal{P} and \mathcal{T} contains $O(m)$ points. If \mathcal{D} contains a monotone juxtaposition, then $|\pi|, |\tau| \in O(m \log m)$ and otherwise, $|\pi|, |\tau| \in O(m^2)$. This gives rise to the two different conditional lower bounds.

Beyond grid-preserving embeddings. First, we modify both π and τ such that any embedding that maps the image of P_1 to the image of T_1 must already be grid-preserving. To that end, we add atomic pairs A_1, A_2 to the initial tile P_1 such that A_1 is to the left and below everything else and A_2 is to the right and above everything else. We then attach to both A_1, A_2 a chain of copy gadgets going all the way to the last tile of the path. We modify \mathcal{T} , in the same way, using chains of copy gadgets originating in the atomic pairs B_1 and B_2 . Observe that in any embedding that sends P_1 to T_1 , the image of A_α is mapped to the image of B_α for each $\alpha \in \{1, 2\}$. The chain of copy gadgets attached to A_α then must map to the chain of gadgets attached to B_α and these chains force the embedding to be grid-preserving.

Finally, we modify π and τ to obtain permutations π' and τ' such that any embedding of π' into τ' can be translated to an embedding of π into τ that maps P_1 to T_1 and vice versa. Let r and q be the lowest and topmost points in P_1 and similarly, let s and t be the lowest and topmost points of T_1 . The family of tiles \mathcal{P}' is obtained from \mathcal{P} by inflating both r and q with an increasing sequence of length $|\tau| + 1$ and similarly, the family \mathcal{T}' is obtained from \mathcal{T} by inflating both s and t . We call the points obtained by inflating r and q *lower anchors* and the ones obtained by inflating s and t *upper anchors*. We let π' and τ' be the \mathcal{F} -assemblies of \mathcal{P}' and \mathcal{T}' . Observe that these modifications did not change the asymptotic size of the input as $|\pi'| = O(|\tau|)$ and $|\tau'| = O(|\tau'|) = O(|\tau|)$.

A.3 Correctness

The “only if” part. Let Φ be a satisfiable formula and fix arbitrary satisfying assignment. We map the image of P_1 to the image of T_1 . In the assignment phase, we map the pair $X_{k,1}^1$ to $Y_{k,1}^1$ if x_k is set to true, otherwise we map it to $Z_{k,1}^1$. The embedding of the multiplication and sorting phase is uniquely determined by the gadgets. It is easy to check that for each satisfied clause K_j there is a way to extend the mapping to the evaluation phase.

The “if” part. Let ϕ be an embedding of π' into τ' . The total length of the anchors in both π' and τ' is $2|\tau| + 2$. Therefore, at least $|\tau| + 2$ points of the anchors in π' must be mapped to the anchors in τ' and in particular, there is at least one point in each anchor of π' that maps to corresponding anchor in τ' . The chains of copy gadgets attached to A_1 and A_2 force the rest of the embedding to be grid-preserving, and thus it straightforwardly translates to a grid-preserving embedding of π into τ .

Using the grid-preserving embedding, we define a satisfying assignment $\rho: [n] \rightarrow \{T, F\}$. We set $\rho(k) = T$ if the pair $X_{k,1}^1$ is mapped to $Y_{k,1}^1$ and we set $\rho(k) = F$ if it is mapped to $Z_{k,1}^1$. This property is clearly maintained throughout the multiplication and sorting phases due to the properties of the gadgets. Finally, we already argued that our construction of evaluation phase guarantees that all three literals in a given clause cannot be negative.

Sets of Linear Forms Which Are Hard to Compute

Michael Kaminski ✉

Department of Computer Science, Technion – Israel Institute of Technology, Haifa, Israel

Igor E. Shparlinski ✉🏠

School of Mathematics and Statistics, University of New South Wales, Sydney, Australia

Abstract

We present a uniform description of sets of m linear forms in n variables over the field of rational numbers whose computation requires $m(n - 1)$ additions. Our result is based on bounds on the height of the annihilating polynomials in the Perron theorem and an effective form of the Lindemann–Weierstrass theorem which is due to Sert (1999).

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Linear algorithms, additive complexity, effective Perron theorem, effective Lindemann–Weierstrass theorem

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.66

Funding *Igor E. Shparlinski*: During this work, the author was supported by Australian Research Council Grant DP180100201.

Acknowledgements The authors would like to thank Michel Waldschmidt for the suggestion to use the work of Sert [11] in their argument.

1 Introduction

Evaluating a set of a linear forms is a natural computation task that frequently appears in both theory and applications. For a matrix

$$\Delta = \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,n} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,n} \\ \vdots & \vdots & & \vdots \\ \delta_{m,1} & \delta_{m,2} & \cdots & \delta_{m,n} \end{pmatrix} \quad (1)$$

and a column vector

$$\mathbf{x} = (x_1, \dots, x_n)^T \quad (2)$$

linear forms are presented as a matrix-vector product

$$\Delta \mathbf{x} = (\delta_{j,1}x_1 + \dots + \delta_{j,n}x_n)_{j=1}^m \quad (3)$$

in which the matrix entries $\delta_{s,t}$ are fixed values and the vector entries x_s are varying inputs and computations are by means of *linear* algorithms. As expected, the complexity of a linear algorithm is its number of additions and we are interested in sets of linear forms of high complexity.

Obviously, the set of linear forms (3) can be computed in $m(n - 1)$ additions. However, in finite fields, this trivial upper bound is not the best possible. Namely, over a finite field of q elements, it can be computed in $O(mn/\log_q m)$ additions, see [10, Theorem 1], where the implied constants are absolute. On the other hand (in finite fields), when $m = O(n)$, there exist $\delta_{s,t}$, $s = 1, \dots, m$ and $t = 1, \dots, n$, for which any computation of (3) requires $\Omega(mn/\log_q m)$ additions, cf. [10, Section 5]. In fact, this lower bound holds for almost all



© Michael Kaminski and Igor E. Shparlinski;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 66; pp. 66:1–66:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$m \times n$ matrices with $m = O(n)$, see Appendix B.2 for the precise statement and the proof by a *counting argument*. Thus, for each pair of positive integers m and n such that $m = O(n)$, the entries of such a matrix can be effectively computed, but to describe them uniformly (in m and n) is a very difficult open problem. Even no example of a non-linear complexity is known from the literature.

The situation is quite different when the underlying field is infinite. By a *transcendence degree argument*, it is easy to see that, over the field of real numbers \mathbb{R} , say, when the coefficients of the linear forms are algebraically independent, the computation of (3) requires $m(n-1)$ additions (cf. [4, Section 5.2]). This leads to a natural question: what about the field of rational numbers \mathbb{Q} ? Refining the transcendence degree argument we establish the existence of sets of linear forms (3) over \mathbb{Q} whose computation requires $m(n-1)$ additions. Moreover, as it is shown in Appendix B.3, almost all sets of linear forms (3) are of such complexity. The main result of our paper is a uniform description of such a set.

Namely, we show that, if the entries of a matrix Δ are algebraically independent and Γ is “sufficiently close” to Δ , then computing $\Gamma\mathbf{x}$ also requires $m(n-1)$ additions. However, an estimate of the above “sufficiently close” and, as a corollary, uniform description of such matrices are based on very non-trivial number-theoretic tools [11] and also involves lengthy and somewhat tedious calculations. Furthermore, we believe that some of our results, such as bounds on the height of the annihilating polynomials in the Perron theorem, are of independent interest and may find further applications.

This paper is organized as follows. In Section 2 we present the definition of a linear algorithm and its associated graph and in Section 3, we normalize linear algorithms and state some simple basic complexity results. In Section 4, we prove the existence of a set of linear forms of the maximal complexity over \mathbb{Q} . Then, in Section 5, we present an alternative proof of the existence of such a set and outline an example. In Section 6 we estimate the height of the polynomial¹ appearing in the alternative proof and in Section 7 we recall an effective version of the *Lindemann-Weierstrass theorem* of Sert [11], which makes the existence proof constructive.² Finally, in Section 8, we present an example of a set of m linear forms in n variables over \mathbb{Q} whose computation requires $m(n-1)$ additions.

To shorten the calculations and to simplify the final result, we use very crude estimates routinely applying inequalities of the form $N^N + N \leq 2N^N$, $e^{N^N} < N^{N^2}$ and similar. Nevertheless these estimation are still sufficient to make the point and to illustrate the approach.

Throughout the paper we adhere the convention that our main results are called “*Theorems*”, our auxiliary results are “*Propositions*”, while all previously known results (regardless of their depth and importance) are called “*Lemmas*”.

2 Linear algorithms and their associated graphs

A *linear algorithm* over a field \mathbb{F} in *indeterminates* x_1, x_2, \dots, x_n consists of a sequence of operations $u_i \leftarrow \alpha_i u_{j_i} + \beta_i u_{k_i}$, $i = 1, \dots, C$, where

- $\alpha_i, \beta_i \in \mathbb{F}^*$ are the algorithm coefficients;
- u_i is the algorithm *variable* that does not appear in a previous step;

¹ That is, the maximum of the absolute values of the polynomial coefficients.

² The Lindemann–Weierstrass theorem states that if algebraic numbers $\alpha_1, \dots, \alpha_N$ are linearly independent over \mathbb{Q} , then $e^{\alpha_1}, \dots, e^{\alpha_N}$ are algebraically independent.

- u_{j_i} and u_{k_i} are either indeterminates or the algorithm variables appearing in a previous step (that is, if u_{j_i} and u_{k_i} are the algorithm variables appearing at step i , then $j_i, k_i < i$).

With each algorithm variable u in a linear algorithm we associate the following linear form $\ell(u)$:

- if u is an indeterminate x_t , then $\ell(u)$ is x_t ;
- if u is the left-hand side of an operation $u \leftarrow \alpha v + \beta w$, then $\ell(u)$ is the linear form $\alpha\ell(v) + \beta\ell(w)$.

A linear algorithm *computes* a linear form $\ell(x_1, \dots, x_n)$, if there is a variable, or an indeterminate, u of the algorithm and a *constant* $\gamma \in \mathbb{F}$ such that $\ell(x_1, \dots, x_n) = \gamma\ell(u)$ (thus, linear algorithms compute linear forms up to scaling by a constant) and a linear algorithm computes a set linear forms

$$\mathcal{L}(x_1, \dots, x_n) = \{\ell_s(x_1, \dots, x_n) : s = 1, \dots, m\}$$

if it computes each form $\ell_s(x_1, \dots, x_n) \in \mathcal{L}(x_1, \dots, x_n)$.

The number n of the variables and the number m of linear forms are fixed throughout this paper.

► **Definition 1.** *The complexity $|\mathcal{A}|$ of a linear algorithm \mathcal{A} is the length C of its sequence of operations.*

► **Definition 2.** *The (additive) complexity of a set of linear forms is the minimal complexity of a linear algorithm that computes the set and algorithms of that minimal complexity are called optimal.*

It is known from [13] that if a set of linear forms over an infinite field can be computed in C additions by a *straight-line* algorithm (see [1, Section 12.2]), then it also can be computed in C additions by a linear algorithm. In other words, multiplications and divisions “cannot replace additions”.

With a linear algorithm \mathcal{A} we associate a labelled directed acyclic graph $G_{\mathcal{A}}(V_{\mathcal{A}}, E_{\mathcal{A}})$, whose set of vertices is the union of $\{x_1, \dots, x_n\}$ and the set of the variables of \mathcal{A} and there is an edge from vertex v to vertex u , if there is an operation of the form $u \leftarrow \alpha v + \beta w$ or the form $u \leftarrow \alpha w + \beta v$. In the former case, the edge is labelled α and, in the latter case, it is labelled β . We denote the label of edge e by $\lambda(e)$.

By definition, $|V_{\mathcal{A}}| = n + |\mathcal{A}|$ and the number of vertices of $G_{\mathcal{A}}$ of the in-degree 2 is $|\mathcal{A}|$.

Let $\pi = e_1, \dots, e_k$ be a path of edges in $G_{\mathcal{A}}$. The *weight* $w(\pi)$ of π is defined, recursively, as follows.

- If π is of length zero, then $w(\pi) = 1$;
- $w(\pi, e) = w(\pi)\lambda(e)$, where π, e is the path π extended with edge e .

The following correspondence between linear algorithms and their associated graphs is well-known from the literature.

► **Lemma 3** (See, e.g., [4, Remark 13.19]). *Let*

$$\mathcal{A} = \{u_i \leftarrow \alpha_i u_{j_i} + \beta_i u_{k_i} : i = 1, \dots, C\}$$

be a linear algorithm and let $\Pi_{\mathcal{A}}(x_t, u_i)$ denote the set of all paths of edges from the indeterminate x_t to the algorithm variable u_i in $G_{\mathcal{A}}$. Then

$$\ell(u_i) = \sum_{t=1}^n \left(\sum_{\pi \in \Pi_{\mathcal{A}}(x_t, u_i)} w(\pi) \right) x_t, \quad i = 1, \dots, C. \quad (4)$$

3 Normalized linear algorithms

In this section we introduce a subclass of linear algorithms called *normalized* linear algorithms. These algorithms have the same computation power, but are more convenient for dealing with complexity issues.

► **Definition 4.** *A linear algorithm is normalized if in each its operation*

$$u_i \leftarrow \alpha_i u_{j_i} + \beta_i u_{k_i}$$

the coefficient α_i of u_{j_i} is 1. The coefficient β_i of u_{k_i} is called a proper coefficient.

We say that a label is *proper* if it is a proper coefficient of the algorithm.

The result below immediately follows from Definition 4 and the definition of the associated graph $G_{\mathcal{A}}$ of an algorithm \mathcal{A} .

► **Proposition 5.** *The additive complexity of a normalized linear algorithm \mathcal{A} equals to the number of proper labels of its associated graph $G_{\mathcal{A}}$.*

Furthermore, we also have the following results.

► **Proposition 6.** *For each linear algorithm there is a normalized linear algorithm of the same complexity that computes the same set of linear forms.*

Proof. Let

$$\mathcal{A} = \{u_i \leftarrow \alpha_i u_{j_i} + \beta_i u_{k_i} : i = 1, \dots, C\}$$

be a linear algorithm. It suffices to show that there exists a normalized linear algorithm

$$\tilde{\mathcal{A}} = \{\tilde{u}_i \leftarrow \tilde{u}_{j_i} + \tilde{\beta}_i \tilde{u}_{k_i} : i = 1, \dots, C\}$$

and constants $\gamma_i \in \mathbb{F}$, $i = 1, \dots, C$, such that $\ell(u_i) = \gamma_i \ell(\tilde{u}_i)$, $i = 1, \dots, C$.

We convert \mathcal{A} to $\tilde{\mathcal{A}}$ and define the constants γ_i by recursion on $i = 1, \dots, C$.

The first addition of \mathcal{A} is $u_1 \leftarrow \alpha_1 x_s + \beta_1 x_t$, implying $\ell(u_1) = \alpha_1 x_s + \beta_1 x_t$. We put

$$\tilde{\beta}_1 = \frac{\beta_1}{\alpha_1} \quad \text{and} \quad \gamma_1 = \alpha_1.$$

Then

$$\ell(u_1) = \alpha_1 x_s + \beta_1 x_t = \alpha_1 \left(x_s + \frac{\beta_1}{\alpha_1} x_t \right) = \gamma_1 (x_s + \tilde{\beta}_1 x_t) = \gamma_1 \ell(\tilde{u}_1).$$

The $(i+1)$ -st addition of \mathcal{A} is $u_{i+1} \leftarrow \alpha_{i+1} u_{j_{i+1}} + \beta_{i+1} u_{k_{i+1}}$, implying

$$\ell(u_{i+1}) = \alpha_{i+1} \ell(u_{j_{i+1}}) + \beta_{i+1} \ell(u_{k_{i+1}}).$$

We put

$$\tilde{\beta}_{i+1} = \frac{\beta_{i+1} \gamma_{k_{i+1}}}{\alpha_{i+1} \gamma_{j_{i+1}}} \quad \text{and} \quad \gamma_{i+1} = \alpha_{i+1} \gamma_{j_{i+1}}.$$

Then

$$\begin{aligned} \ell(u_{i+1}) &= \alpha_{i+1} \ell(u_{j_{i+1}}) + \beta_{i+1} \ell(u_{k_{i+1}}) \\ &= \alpha_{i+1} \gamma_{j_{i+1}} \ell(\tilde{u}_{j_{i+1}}) + \beta_{i+1} \gamma_{k_{i+1}} \ell(\tilde{u}_{k_{i+1}}) \\ &= \alpha_{i+1} \gamma_{j_{i+1}} \ell\left(\tilde{u}_{j_{i+1}} + \frac{\beta_{i+1} \gamma_{k_{i+1}}}{\alpha_{i+1} \gamma_{j_{i+1}}} \tilde{u}_{k_{i+1}}\right) \\ &= \gamma_{i+1} \ell(\tilde{u}_{j_{i+1}} + \tilde{\beta}_{i+1} \tilde{u}_{k_{i+1}}) \\ &= \gamma_{i+1} \ell(\tilde{u}_{i+1}), \end{aligned}$$

which concludes the proof. ◀

From now on, we assume that all linear algorithms under consideration are over \mathbb{R} and by Proposition 6, we assume that they are normalized.

We represent a linear form $\ell(x_1, \dots, x_n) = \sum_{t=1}^n \delta_t x_t$ by the product $(\delta_1, \dots, \delta_n)\mathbf{x}$, where \mathbf{x} is the (column) vector of the indeterminates x_1, \dots, x_n as in (2). Similarly, we represent a set of linear forms

$$\ell_s(x_1, \dots, x_n) = \sum_{t=1}^n \delta_{s,t} x_t, \quad s = 1, \dots, m,$$

by a matrix-vector product $\Delta\mathbf{x}$, where the s th row of the matrix Δ is the row vector $(\delta_{s,1}, \dots, \delta_{s,n})$ of the coefficients of $\ell_s(x_1, \dots, x_n)$, see (3).

4 Computation of linear forms over \mathbb{R} and \mathbb{Q}

In this section we prove the existence of a matrix $\Delta \in \mathbb{Q}^{m \times n}$ such that computing the set of linear forms $\Delta\mathbf{x}$ requires $m(n-1)$ additions. As a preliminary step, we consider matrices with entries from \mathbb{R} .

► **Theorem 7** (Cf. [4, Theorem 13.10]). *If all the coefficients of the linear forms (3) are algebraically independent, then the additive complexity of (3) is $m(n-1)$.*³

Proof. Let \mathcal{A} be a linear algorithm that computes (3) and let $G_{\mathcal{A}} = (V_{\mathcal{A}}, E_{\mathcal{A}})$ be its associated labelled graph.

Let u_{i_s} and γ_s , $s = 1, \dots, m$, be the algorithm variables and the respective constants such that

$$\ell_s(x_1, \dots, x_n) = \sum_{t=1}^n \delta_{s,t} x_t = \gamma_s \ell(u_{i_s}).$$

Then, by Lemma 3,

$$\delta_{s,t} = \gamma_s \sum_{\pi \in \Pi_{\mathcal{A}}(x_t, u_{i_s})} w(\pi) = P_{s,t}(\gamma_s, \beta_1, \dots, \beta_{|\mathcal{A}|})$$

for some polynomials $P_{s,t}(Y_s, X_1, \dots, X_{|\mathcal{A}|})$, $s = 1, \dots, m$ and $t = 1, \dots, n$.

If the number $|\mathcal{A}|$ of the proper labels is less than $m(n-1)$, then the total number of β and γ variables is less than mn , implying that these mn polynomials are algebraically dependent, see Proposition 5.

Let $P(Z_{1,1}, \dots, Z_{m,n})$ be a polynomial over \mathbb{Q} such that

$$P(P_{1,1}(Y_1, X_1, \dots, X_{|\mathcal{A}|}), \dots, P_{m,n}(Y_m, X_1, \dots, X_{|\mathcal{A}|})) = 0. \quad (5)$$

Then

$$P(P_{1,1}(\gamma_1, \beta_1, \dots, \beta_{|\mathcal{A}|}), \dots, P_{m,n}(\gamma_m, \beta_1, \dots, \beta_{|\mathcal{A}|})) = 0,$$

implying

$$P(\delta_{1,1}, \dots, \delta_{m,n}) = 0. \quad (6)$$

That is, the matrix entries $\delta_{s,t}$ are algebraically dependent, in contradiction with the condition of the theorem. ◀

³ In [4] this result is attributed to [16], but the proof in [16] is very implicit.

66:6 Sets of Linear Forms Which Are Hard to Compute

Recall that we aim to find an $m \times n$ matrix over \mathbb{Q} that defines the set of linear forms of complexity $m(n-1)$. The existence of such a matrix is shown below (see Theorem 15), but to describe its entries is not straightforward at all and we do this in Section 8.

To proceed with our proof of existence we first need to introduce a number of definitions and auxiliary results.

► **Definition 8.** We say that $P(Z_1, \dots, Z_N) \in \mathbb{Z}[Z_1, \dots, Z_N]$ is an annihilating polynomial of $P_k(X_1, \dots, X_{N-1}) \in \mathbb{Z}[X_1, \dots, X_{N-1}]$, $k = 1, \dots, N$, if P is a non-zero polynomial and

$$P(P_1(X_1, \dots, X_{N-1}), \dots, P_N(X_1, \dots, X_{N-1})) = 0.$$

► **Definition 9.** We say that the polynomial $P_{s,t}(Y_s, X_1, \dots, X_{|\mathcal{A}|})$, $s = 1, \dots, m$ and $t = 1, \dots, n$, defined in the proof of Theorem 7 is associated with the (s, t) -th entry of Δ via \mathcal{A} .

► **Definition 10.** We say that a polynomial P is (m, n) -associated if for some $m \times n$ matrix Δ , some $s = 1, \dots, m$, some $t = 1, \dots, n$ and some algorithm \mathcal{A} with $|\mathcal{A}| < m(n-1)$ that computes (3), P is associated with the (s, t) -th entry of Δ via \mathcal{A} .

► **Proposition 11.** The set of (m, n) -associated polynomials is finite.

Proof. By (4), the coefficients of $P_{s,t}(Y_s, X_1, \dots, X_{|\mathcal{A}|})$ are zero or one and, by definition, $|\mathcal{A}| < m(n-1)$. Therefore

$$\deg P_{s,t}(Y_s, X_1, \dots, X_{|\mathcal{A}|}) < m(n-1),$$

where $s = 1, \dots, m$ and $t = 1, \dots, n$, and the result follows. ◀

► **Definition 12.** The polynomial $P(Z_{1,1}, \dots, Z_{m,n}) \in \mathbb{Q}[Z_{1,1}, \dots, Z_{m,n}]$ satisfying (5), that is, an annihilating polynomial of

$$P_{1,1}(Y_1, X_1, \dots, X_{|\mathcal{A}|}), \dots, P_{m,n}(Y_m, X_1, \dots, X_{|\mathcal{A}|}),$$

is called an \mathcal{A} - Δ -annihilating polynomial.

► **Definition 13.** We say that a set of polynomials \mathfrak{Q} is (m, n) -complete if for each $m \times n$ matrix Δ and each algorithm \mathcal{A} with $|\mathcal{A}| < m(n-1)$ that computes (3), \mathfrak{Q} contains an \mathcal{A} - Δ -annihilating polynomial.

For our specific example we shall look for a common non-zero of all polynomials in an (m, n) -complete set. This resembles the approach in [14], see also [4, Section 9.4].

► **Proposition 14.** Let \mathfrak{Q} be a minimal with respect to inclusion (m, n) -complete set of polynomials. Then \mathfrak{Q} is finite.

Proof. This is because number of polynomials in \mathfrak{Q} does not exceed the number of all k -tuples, $k < m(n-1)$ of (m, n) -associated polynomials and by Proposition 11 the number of such tuples is finite. ◀

In what follows, for a matrix A we denote by $|A|$ its *Frobenius norm*, that is, the square root of the sum of squares of its entries. Similarly, for a vector α we use $|\alpha|$ to denote its *Euclidean norm*.

► **Theorem 15.** There exists an $m \times n$ matrix over \mathbb{Q} that defines the set of linear forms of complexity $m(n-1)$.

Proof. Let $\Delta_1, \Delta_2, \dots$, be a sequence of $m \times n$ matrices over \mathbb{Q} that converges (in the Frobenius norm) to the matrix $\Delta = (\delta_{s,t})$ from the proof of Theorem 7. That is, $\lim_{i \rightarrow \infty} |\Delta_i - \Delta| = 0$.

Let $\Delta_i = (\delta_{s,t,i})$, $i = 1, 2, \dots$. Then

$$\lim_{i \rightarrow \infty} \delta_{s,t,i} = \delta_{s,t} \quad s = 1, \dots, m \text{ and } t = 1, \dots, n. \quad (7)$$

Assume to the contrary that for each $i = 1, 2, \dots$, there is an algorithm \mathcal{A}_i , $|\mathcal{A}_i| < m(n-1)$, that computes $\Delta_i \mathbf{x}$. Let Ω be a finite (m, n) -complete set provided by Proposition 14. Then, like in the proof of Theorem 7, it can be shown that for each $i = 1, 2, \dots$, there is an \mathcal{A}_i - Δ_i -annihilating polynomial $P_i(Z_{1,1}, \dots, Z_{m,n}) \in \Omega$ such that $P_i(\delta_{1,1,i}, \dots, \delta_{m,n,i}) = 0$.

Thus, there is a subsequence $\tilde{\Delta}_1, \tilde{\Delta}_2, \dots$ of $\Delta_1, \Delta_2, \dots$ with the same annihilating polynomial $P(Z_{1,1}, \dots, Z_{m,n})$, implying $P(\tilde{\delta}_{1,1,i}, \dots, \tilde{\delta}_{m,n,i}) = 0$, $i = 1, 2, \dots$, which, by (7), implies (6). However, the latter contradicts the algebraic independence of the entries of Δ . \blacktriangleleft

5 An alternative proof of Theorem 15 and an outline of an example

Here we present an alternative and more constructive approach of the existence of a set of m linear forms in n variables over \mathbb{Q} of complexity $m(n-1)$ and outline our example. To simplify the calculations, we look for a matrix Γ defining such a set in the ball of radius 1 centered at Δ . We show in Section 7 that this condition is redundant.

Theorem 16 below is in the background of our example of linear forms which are hard to compute.

► **Theorem 16.** *Let Ω be an (m, n) -complete set of polynomials. Let $\Gamma = (\gamma_{s,t})$ and $\Delta = (\delta_{s,t})$ be $m \times n$ matrices, where $\delta_{s,t}$, $s = 1, \dots, m$ and $t = 1, \dots, n$, are algebraically independent and $|\Delta - \Gamma| < 1$. Let $\gamma = (\gamma_{1,1}, \dots, \gamma_{m,n})$ and let $\delta = (\delta_{1,1}, \dots, \delta_{m,n})$. If*

$$|\delta - \gamma| < \min_{Q \in \Omega} \min_{|\mu| < 1} \left\{ \frac{|Q(\delta)|}{|\nabla Q(\delta + \mu)|} \right\}, \quad (8)$$

then computing $\Gamma \mathbf{x}$ requires $m(n-1)$ additions.

The proof of Theorem 16 is based on the following lemma.

► **Lemma 17** (The mean value theorem for several variables, see, e.g., [2, Chapter 12, Example 1]). *Let $F : \mathbb{R}^N \rightarrow \mathbb{R}$ be a differentiable function and $\alpha_1, \alpha_2 \in \mathbb{R}^N$. Then, for some $\vartheta \in [0, 1]$,*

$$F(\alpha_2) - F(\alpha_1) = \nabla F(\alpha_1 + \vartheta(\alpha_2 - \alpha_1)) \cdot (\alpha_2 - \alpha_1),$$

where ∇F denotes the gradient of F .

Proof of Theorem 16. Assume to the contrary that the set of linear forms $\Gamma \mathbf{x}$ can be computed in less than $m(n-1)$ additions by an algorithm \mathcal{A} . Then, since Ω is (m, n) -complete, there is an \mathcal{A} - Γ -annihilating polynomial $Q \in \Omega$ such that $Q(\gamma) = 0$.

By Lemma 17, for some $\vartheta \in [0, 1]$

$$Q(\delta) - Q(\gamma) = \nabla Q(\gamma + \vartheta(\delta - \gamma)) \cdot (\delta - \gamma). \quad (9)$$

Since

$$\gamma + \vartheta(\delta - \gamma) = \vartheta\delta + (1 - \vartheta)\gamma = \vartheta\delta + (1 - \vartheta)(\delta + \gamma - \delta) = \delta + (1 - \vartheta)(\gamma - \delta)$$

66:8 Sets of Linear Forms Which Are Hard to Compute

and $|(1 - \vartheta)(\gamma - \delta)| < 1$, by (9), for $\mu = (1 - \vartheta)(\gamma - \delta)$, we have

$$Q(\delta) = \nabla Q(\delta + \mu) \cdot (\delta - \gamma) \quad (10)$$

because $Q(\gamma) = 0$. It follows from (10), by the Cauchy–Schwarz inequality, that

$$|Q(\delta)| \leq |\nabla Q(\delta + \mu)| |\delta - \gamma|,$$

which contradicts (8). ◀

Note that, since the components of δ are algebraically independent, $Q(\delta) \neq 0$. Thus, for a finite (m, n) -complete set of polynomials Ω , the right-hand side of (8) is positive.

Using the trivial inequality

$$|\delta - \gamma| \leq \sqrt{mn} \max\{|\delta_{s,t} - \gamma_{s,t}| : s = 1, \dots, m, t = 1, \dots, n\}$$

we see that Theorem 16 yields the corollary below.

► **Corollary 18.** *Let Ω be an (m, n) -complete set of polynomials. Let $\Gamma = (\gamma_{s,t})$ and $\Delta = (\delta_{s,t})$ be $m \times n$ matrices, where $\delta_{s,t}$, $s = 1, \dots, m$ and $t = 1, \dots, n$, are algebraically independent and $|\Delta - \Gamma| < 1$. Let $\gamma = (\gamma_{1,1}, \dots, \gamma_{m,n})$ and $\delta = (\delta_{1,1}, \dots, \delta_{m,n})$. If*

$$\max\{|\delta_{s,t} - \gamma_{s,t}| : s = 1, \dots, m \text{ and } t = 1, \dots, n\} < \frac{1}{\sqrt{mn}} \min_{Q \in \Omega} \min_{|\mu| < 1} \left\{ \frac{|Q(\delta)|}{|\nabla Q(\delta + \mu)|} \right\},$$

then computing $\Gamma \mathbf{x}$ requires $m(n - 1)$ additions.

Following the above existence proof, for an example of a rational $m \times n$ matrix defining a set of linear forms of complexity $m(n - 1)$ we need a set of algebraically independent numbers $\delta_{s,t}$, $s = 1, \dots, m$ and $t = 1, \dots, n$, for which the right-hand side of (8) can be effectively estimated. Such a set is provided by an effective version of the Lindemann-Weierstrass theorem due to Sert [11], see Lemma 26 in Section 7.

6 The degree and the height of annihilating polynomials

In this section we estimate the denominator of the right-hand side of (8). For this we need to estimate the degree and the height of (m, n) -polynomials which are defined as follows.

► **Definition 19.** *A polynomial P is called an (m, n) -polynomial if for some $m \times n$ matrix Δ and some algorithm \mathcal{A} , $|\mathcal{A}| < m(n - 1)$, that computes (3), P is \mathcal{A} - Δ -annihilating.*

Recall that these polynomials arise from linear algorithms of the complexity less than $m(n - 1)$ and satisfy (5).

We start with the following result, that is essentially due to Perron.

► **Lemma 20** ([8], see also [9, Theorem 1.1] for a self-contained proof.). *Let*

$$P_k(X_1, \dots, X_{N-1}) \in \mathbb{Z}[X_1, \dots, X_{N-1}]$$

with $\deg P_k(X_1, \dots, X_{N-1}) = d_k$, $k = 1, \dots, N$. Then there exists an annihilating polynomial $P(Z_1, \dots, Z_N) \in \mathbb{Z}[Z_1, \dots, Z_N]$ of P_1, \dots, P_N such that

$$\deg P \leq \frac{d_1 \times \dots \times d_N}{\min\{d_1, \dots, d_N\}}.$$

Actually, in [8], the polynomial $P(Z_1, \dots, Z_N)$ has rational coefficients, but, multiplying them by their common denominator, we obtain a polynomial over \mathbb{Z} .

In the case of (m, n) -polynomials, $N = mn$ and $d_s \leq m(n-1)$, $i = 1, \dots, N$. Therefore, by Lemma 20, we may assume that the degree of (m, n) -polynomials under consideration does not exceed N^{N-1} .

Next, we are going to estimate the minimum height $h(P)$ of the polynomial $P(Z_1, \dots, Z_N)$ provided by Lemma 20. This can be done by solving a system of linear homogeneous equations, see [14, Lemma 2.2], [4, Lemma 9.28] or [9, Property 1.2].

We need some auxiliary results first. We start with recalling the following well-known upper bound on the height of a polynomial product.

► **Lemma 21** (See, e.g., [5, Lemma 1.2(1)(b)], where the logarithmic height $\ln h(P)$ is used.). Let $P_k \in \mathbb{Z}[X_1, \dots, X_{N-1}]$, $k = 1, \dots, \ell$. Then

$$h\left(\prod_{k=1}^{\ell} P_k\right) \leq N^{\sum_{k=1}^{\ell} \deg P_k} \prod_{k=1}^{\ell} h(P_k).$$

We also recall the classical *Siegel lemma*.

► **Lemma 22** ([12, Page 213, Hilfssatz], see also [3, 15] for further improvements and generalizations.). If a system of J linear homogeneous equations in $I > J$ variables

$$\sum_{i=1}^I b_{i,j} z_i = 0, \quad j = 1, \dots, J,$$

with $B = (b_{i,j})_{i,j=1}^{I,J} \in \mathbb{Z}^{I \times J}$, has a nonzero solution, then it has a nonzero integer solution $\mathbf{v} = (v_1, \dots, v_I)$ with

$$h(\mathbf{v}) \leq 1 + (Ih(B))^{J/(I-J)},$$

where

$$h(B) = \max\{|b_{i,j}| : i = 1, \dots, I, j = 1, \dots, J\}$$

and

$$h(\mathbf{v}) = \max\{|v_i| : i = 1, \dots, I\}. \quad (11)$$

We are now ready to estimate the height of an annihilating polynomial. More precisely we establish the following result.

► **Proposition 23.** Let $P(Z_1, \dots, Z_N) \in \mathbb{Z}[Z_1, \dots, Z_N]$ be an annihilating polynomial of

$$P_k(X_1, \dots, X_{N-1}) \in \mathbb{Z}[X_1, \dots, X_{N-1}], \quad k = 1, \dots, N.$$

There exists another annihilating polynomial $Q(Z_1, \dots, Z_N) \in \mathbb{Z}[Z_1, \dots, Z_N]$ of P_1, \dots, P_N of degree and height

$$\deg Q \leq \deg P,$$

$$h(Q) \leq 1 + \left(\binom{\deg P + N}{N} N^{d_{\max} \deg P} h_{\max}^{\deg P} \right)^{\binom{\deg P + N}{N} - 1},$$

respectively, where

$$d_{\max} = \max\{\deg P_k : k = 1, \dots, N\},$$

$$h_{\max} = \max\{h(P_k) : k = 1, \dots, N\}.$$

66:10 Sets of Linear Forms Which Are Hard to Compute

Proof. We employ the following notation:

- $\mathbf{X} = (X_1, \dots, X_{N-1})$ and $\mathbf{Z} = (Z_1, \dots, Z_N)$ are vectors of variables;
- $\mathbf{i} = (i_1, \dots, i_N)$ and $\mathbf{j} = (j_1, \dots, j_{N-1})$ are vectors of non-negative integers;
- $\mathbf{X}^{\mathbf{j}} = \prod_{s=1}^{N-1} X_s^{j_s}$ and $\mathbf{Z}^{\mathbf{i}} = \prod_{k=1}^N Z_k^{i_k}$ are multivariate monomials.

We search for a polynomial Q in the form

$$Q(Z_1, \dots, Z_N) = \sum_{\mathbf{i}: i_1 + \dots + i_N \leq \deg P} v_{\mathbf{i}} \mathbf{Z}^{\mathbf{i}}, \quad (12)$$

with unknown coefficients $v_{\mathbf{i}}$ to be determined.

To find the coefficients $v_{\mathbf{i}}$ of $Q(Z_1, \dots, Z_N)$, we substitute the polynomials $P_k(X_1, \dots, X_{N-1})$ for Z_k , $k = 1, \dots, N$, in (12), obtaining

$$\begin{aligned} Q(P_1(X_1, \dots, X_{N-1}), \dots, P_N(X_1, \dots, X_{N-1})) \\ = \sum_{\mathbf{i}: i_1 + \dots + i_N \leq \deg P} v_{\mathbf{i}} \prod_{k=1}^N P_k^{i_k}(X_1, \dots, X_{N-1}) = 0. \end{aligned} \quad (13)$$

Let

$$\prod_{k=1}^N P_k^{i_k}(X_1, \dots, X_{N-1}) = \sum_{\mathbf{j}: j_1 + \dots + j_{N-1} \leq d_{\max} \deg P} \mathbf{c}_{\mathbf{i}, \mathbf{j}} \mathbf{X}^{\mathbf{j}}.$$

Then, it follows from (13) that

$$\begin{aligned} \sum_{\mathbf{i}: i_1 + \dots + i_N \leq \deg P} v_{\mathbf{i}} \sum_{\mathbf{j}: j_1 + \dots + j_{N-1} \leq d_{\max} \deg P} \mathbf{c}_{\mathbf{i}, \mathbf{j}} \mathbf{X}^{\mathbf{j}} \\ = \sum_{\mathbf{j}: j_1 + \dots + j_{N-1} \leq d_{\max} \deg P} \mathbf{X}^{\mathbf{j}} \left(\sum_{\mathbf{i}: i_1 + \dots + i_N \leq \deg P} \mathbf{c}_{\mathbf{i}, \mathbf{j}} v_{\mathbf{i}} \right) = 0. \end{aligned}$$

That is, we have a system of linear homogeneous equations

$$\sum_{\mathbf{i}: i_1 + \dots + i_N \leq \deg P} \mathbf{c}_{\mathbf{i}, \mathbf{j}} v_{\mathbf{i}} = 0, \quad \mathbf{j}: j_1 + \dots + j_{N-1} \leq d_{\max} \deg P \quad (14)$$

in

$$I = \binom{\deg P + N}{N} \quad (15)$$

unknowns $v_{\mathbf{i}}$ (the coefficients of $P(Z_1, \dots, Z_N)$).

We also note that for the coefficients $\mathbf{c}_{\mathbf{i}, \mathbf{j}}$ of the system of linear equations (14) we have

$$\max_{\mathbf{i}, \mathbf{j}} |\mathbf{c}_{\mathbf{i}, \mathbf{j}}| \leq \max_{i_1 + \dots + i_N \leq \deg P} h \left(\prod_{k=1}^N P_k^{i_k} \right),$$

where \mathbf{i} and \mathbf{j} run through the vectors with $i_1 + \dots + i_N \leq \deg P$ and $j_1 + \dots + j_{N-1} \leq d_{\max} \deg P$, respectively

Hence, by Lemma 21 with $\ell = \deg P$,

$$\max_{\mathbf{i}, \mathbf{j}} |\mathbf{c}_{\mathbf{i}, \mathbf{j}}| \leq N^{d_{\max} \deg P} h_{\max}^{\deg P}. \quad (16)$$

Since, by our assumption on the polynomial Q , this system has a non-zero solution, we can select at most $J \leq I - 1$ linear equations forming a system of linear homogeneous equations, which is equivalent to (14). Hence combining the bound (16) with Lemma 22, we derive that (14) has a solution with

$$\max\{v_i : \mathbf{i} = (i_1, \dots, i_N) \text{ with } i_1 + \dots + i_N \leq \deg P\} \leq 1 + (IN^{d_{\max} \deg P} h_{\max}^{\deg P})^{I-1},$$

where I is given by (15), which concludes the proof. \blacktriangleleft

In what follows we renumber the (m, n) -associated polynomials $P_{s,t}$, $s = 1, \dots, m$ and $t = 1, \dots, n$, in (5) as P_1, \dots, P_N , that is, we write

$$\{P_1, \dots, P_N\} = \{P_{s,t} : s = 1, \dots, m, t = 1, \dots, n\}. \quad (17)$$

We remark that by Lemma 20, we can assume $\deg P \leq d_{\max}^{N-1}$ in the notation of Proposition 23. Furthermore, as we have noticed in the proof of Proposition 11, in the special case of (m, n) -associated polynomials we have

$$d_{\max} = N \quad \text{and} \quad h_{\max} = 1. \quad (18)$$

Thus, by Lemma 20, we can assume

$$\deg P \leq N^{N-1}. \quad (19)$$

► **Corollary 24.** *For any $N = mn \geq 4$ and polynomials (17), there exists an (m, n) -polynomial $Q(Z_1, \dots, Z_N)$ of degree and height satisfying*

$$\deg Q \leq N^{N-1} \quad \text{and} \quad h(Q) \leq N^{2N^{N^2}}, \quad (20)$$

respectively.

Proof. The bound on the degree follows directly from (19). Using the well known estimate

$$\binom{q}{r} \leq \frac{q^r}{r!} \leq (eq/r)^r,$$

which holds for arbitrary integers $q \geq r \geq 1$, we derive

$$\binom{N^{N-1} + N}{N} \leq (e(N^{N-2} + 1))^N = e^N N^{N(N-2)} (1 + 1/N^{N-2})^N \leq e^{N+1} N^{N^2-2N}.$$

Since for $N \geq 4$ we have $e^{N+1} < N^N$, we now obtain

$$\binom{N^{N-1} + N}{N} < N^{N^2-N}.$$

Substituting (18) and (19) in Proposition 23 and using the above estimate, we see that

$$h(Q) \leq 1 + \left(N^{N^N + N^2 - N}\right)^{N^{N^2-N}} = 1 + N^{(N^N + N^2 - N)N^{N^2-N}}.$$

We now use the crude estimate $N^N + N^2 - N < 2N^N$ and obtain

$$h(Q) < 1 + N^{2N^{N^2}}.$$

Since $h(Q)$ is an integer, this concludes the proof. \blacktriangleleft

66:12 Sets of Linear Forms Which Are Hard to Compute

Let $\mathfrak{Q}_{m,n}$ denote the class of annihilating (m,n) -polynomials Q with the degree and height satisfying 20, where $N = mn$. By Corollary 24, we see that for $N = mn \geq 4$, $\mathfrak{Q}_{m,n} \neq \emptyset$. Clearly, $\mathfrak{Q}_{m,n}$ is (m,n) -complete and, therefore, Corollary 18 can be applied with $\mathfrak{Q} = \mathfrak{Q}_{m,n}$.

We remark that the case of $N = mn \leq 3$ is trivial, as then $m = 1$ or $n = 1$. Hence the condition $N \geq 4$, which stems from Corollary 24, is not restrictive.

We also recall the definition of the naive height in (11).

► **Corollary 25.** *For any $N = mn \geq 4$ and $\delta \in \mathbb{R}^N$, we have*

$$\max_{Q \in \mathfrak{Q}_{m,n}} \max_{|\mu| < 1} |\nabla Q(\delta + \mu)| < N^{3N^{N^2}} (h(\delta) + 1)^{N^{N-1}}.$$

Proof. First we estimate

$$\frac{\partial Q}{\partial Z_k}(\delta + \mu), \quad k = 1, \dots, N.$$

The polynomial $\partial Q / \partial Z_k$ is of degree

$$\deg \partial Q / \partial Z_k < \deg Q \leq N^{N-1}$$

and thus, by Corollary 24, of height

$$h(\partial Q / \partial Z_k) \leq N^{N-1} h(Q) \leq N^{2N^{N^2} + N - 1}.$$

The number of monomials in N variables of degree less than N^{N-1} is less than $N^{N^2 - N}$. Thus, for $|\mu| \leq 1$ we have

$$\begin{aligned} \left| \frac{\partial Q}{\partial Z_k}(\delta + \mu) \right| &\leq N^{N^2 - N} h(\partial Q / \partial Z_k) (h(\delta) + 1)^{N^{N-1}} \\ &\leq N^{2N^{N^2} + N^2 - 1} (h(\delta) + 1)^{N^{N-1}} < N^{3N^{N^2}} (h(\delta) + 1)^{N^{N-1}}, \end{aligned}$$

and the result follows. ◀

7 Effective Lindemann–Weierstrass theorem

In this section we estimate the numerator of the right-hand side of (8). This, together with the estimation of the denominator of the right-hand side of (8) in the previous section, would make Theorem 16 constructive. The estimation of the numerator is based on an effective version of the Lindemann–Weierstrass theorem due to Sert [11] stated below. We precede the statement of the theorem with the necessary definitions and notations.

- K is a number field of degree D .
- M_K is the set of normalized absolute values of K , that is, the extensions onto K all the values on \mathbb{Q} (Archimedean and p -adic).
- The *absolute height* $h_a(\alpha)$ of an s -tuple $\alpha = (\alpha_1, \dots, \alpha_s) \in K^s$ is defined by

$$h_a(\alpha) = \prod_{\nu \in M_K} \max_{1 \leq k \leq s} \{1, |\alpha_k|_\nu\}^{D_\nu / D},$$

where D_ν is the local degree of K_ν , that is, the dimension of the ν -completion of K over \mathbb{R} , if ν is Archimedean, or the ν -completion of K over \mathbb{Q}_p , if ν is p -adic.

- Let $\{\beta_1, \dots, \beta_L\}$ be the set of all coefficients of a polynomial $P(x_1, \dots, x_N)$ over K . We denote by Δ_P the discriminant of $\mathbb{Q}(\beta_1, \dots, \beta_L)$.

- The *absolute height* of a multi-variate polynomial P is the absolute height of the tuple of the polynomial coefficients.

Our main technical tool is the following result of Sert.

► **Lemma 26** ([11, Theorem 3]). *Let $P \in K[Z_1, \dots, Z_N]$ be of degree $d \geq 1$ and of absolute height H and let $\alpha = (\alpha_1, \dots, \alpha_N) \in K^N$ be linearly independent over \mathbb{Q} . Then*

$$\begin{aligned} & \ln |P(e^{\alpha_1}, \dots, e^{\alpha_N})| \\ & \geq -c_0 d^N \left(\ln H + \frac{39}{328D} \ln |\Delta_P| + \exp(c_1 d^N + c_2 d^N \ln d + 72 \max\{1, h_a(\alpha)\}) \right), \end{aligned}$$

where

$$\begin{aligned} c_0 &= 41 \times 3^{2N} 2^{-N+1} D^{N+1} N^N, \\ c_1 &= 2^{2-N} 3^{2N+1} D^{N+1} N^N + (1 + 6D) 2^{4-N} 3^{2N} D^N N^N \ln(9DN) \\ & \quad + 2^{4-N} 3^{2N} D^{N+1} N^N (1 + 6N) \ln h_a(\alpha), \\ c_2 &= (1 + 6D) 2^{4-N} 3^{2N} D^N N^N. \end{aligned}$$

► **Remark 27.** As noticed by Sert [11, Page 100], if the coefficients of $P(Z_1, \dots, Z_N)$ are rational integers, then, in Lemma 26, we replace H with the ordinary height $h(P)$ of P and replace $\ln |\Delta_P|$ with zero.

The result below follows from Lemma 26 and Remark 27 by a straightforward substitution.

► **Corollary 28.** *Let $Q(Z_1, \dots, Z_N) \in \mathfrak{Q}_{m,n}$ and let $\alpha_1, \dots, \alpha_N$ be algebraic numbers linearly independent over \mathbb{Q} . Then, for $N \geq 4$,*

$$|Q(e^{\alpha_1}, \dots, e^{\alpha_N})| > \exp\left(-N^{2^{5N}} D^{N+1} N^{N^2+2(D+\max\{1, h_a(\alpha)\})}\right)$$

where D is the degree of the number field $\mathbb{Q}(\alpha_1, \dots, \alpha_N)$.

The proof of this corollary is presented in Appendix A.1.

Now, the substitution of the bound of Corollaries 28 and 25 for the numerator and the denominator of the right-hand side of (8), respectively, makes Theorem 16 constructive.

Finally, for the uniform example of a set of m linear forms in n variables over \mathbb{Q} of complexity $m(n-1)$ in the next section we need the estimation below.

► **Proposition 29.** *Let α be a positive integer such that $[\mathbb{Q}(\alpha^{1/N}) : \mathbb{Q}] = N$, where $N = mn \geq 4$ and $N > \alpha > 1$, and let $\gamma_{s,t}$, $s = 1, \dots, m$ and $t = 1, \dots, n$, be such that*

$$\left| e^{\alpha^{(s-1)n+t)/N}} - \gamma_{s,t} \right| < \exp\left(-N^{3N^2}\right). \quad (21)$$

Then, for $\Gamma = (\gamma_{s,t})$, computing $\Gamma \mathbf{x}$ requires $m(n-1)$ additions.

The proof of this proposition is presented in Appendix A.2.

8 An example

An example of an integer matrix $\Omega \in \mathbb{Z}^{m \times n}$ such that computing $\Omega \mathbf{x}$ requires $m(n-1)$ additions is based on Proposition 29 with $\alpha = 2$. We precede the example with a series of auxiliary calculations for which we need the propositions below.

Throughout we assume that $N \geq 4$.

We recall the definition of binomial coefficients for real arguments:

$$\binom{r}{i} = \frac{r(r-1) \cdots (r-i+1)}{i!}.$$

66:14 Sets of Linear Forms Which Are Hard to Compute

► **Proposition 30.** For $r \in (0, 1)$ and a non-negative integer j ,

$$\left| 2^r - \sum_{i=0}^j \binom{r}{i} \right| < \frac{1}{j+1}$$

Proof. For $x > 0$, the Taylor expansion (with the Lagrange remainder) of $(1+x)^r$ is

$$(1+x)^r = \sum_{i=0}^j \binom{r}{i} x^i + \binom{r}{j+1} \Theta^{j+1} \quad (22)$$

for some $\Theta \in (0, x)$. Substituting 1 for both x and Θ in (22), we obtain

$$\left| 2^r - \sum_{i=0}^j \binom{r}{i} \right| < \left| \binom{r}{j+1} \right| = \frac{|r(r-1)(r-2)\cdots(r-j)|}{(j+1)!} < \frac{j!}{(j+1)!} < \frac{1}{j+1},$$

and the result follows. ◀

By Proposition 30, we approximate $2^{((s-1)n+t)/N}$ by

$$r_{s,t} = \sum_{i=0}^j \binom{((s-1)n+t)/N}{i} \quad s = 1, \dots, m, \quad t = 1, \dots, n,$$

for an appropriate j , to be chosen later.

► **Proposition 31.** For $x \in (1, 2)$ and $k \geq 6$,

$$e^x - \sum_{i=0}^k \frac{x^i}{i!} < \frac{1}{k+1}.$$

Proof. For $x > 0$, the Taylor expansion (with the Lagrange remainder) of e^x is

$$e^x = \sum_{i=0}^k \frac{x^i}{i!} + \frac{\Theta^{k+1}}{(k+1)!}$$

for some $\Theta \in (0, x)$. Thus, using $k! \geq 2^{k+1}$ for $k \geq 6$, we obtain

$$e^x - \sum_{i=0}^k \frac{x^i}{i!} = \frac{\Theta^{k+1}}{(k+1)!} < \frac{2^{k+1}}{(k+1)!} < \frac{1}{k+1},$$

which concludes the proof. ◀

By Proposition 31, we approximate $e^{r_{s,t}}$ by

$$\gamma_{s,t} = \sum_{i=0}^k \frac{r_{s,t}^i}{i!}, \quad s = 1, \dots, m, \quad t = 1, \dots, n,$$

for an appropriate k , to be chosen later.

Our last auxiliary estimation is as follows.

► **Proposition 32.** Let $x \leq 2$ and $\varepsilon \in (-1, 1)$. Then

$$|e^x - e^{x+\varepsilon}| < 42\varepsilon.$$

Proof. By Lemma 17,

$$|e^x - e^{x+\varepsilon}| < \left| e^{x-|\varepsilon|} - e^{x+|\varepsilon|} \right| = 2\varepsilon e^{x+\Theta}$$

for some $\Theta \in (-\varepsilon, \varepsilon)$. Thus, $|e^x - e^{x+\varepsilon}| < 2e^3\varepsilon < 42\varepsilon$. \blacktriangleleft

To simplify the expressions in our example, recalling the bound of Proposition 29 it is now convenient to denote

$$E = \exp\left(N^{3N^2}\right). \quad (23)$$

We contend that for

$$j \geq 126E \quad \text{and} \quad k \geq 3E \quad (24)$$

we have the inequality

$$\left| e^{2^{((s-1)n+t)/N}} - \gamma_{s,t} \right| < \frac{2}{3E}. \quad (25)$$

Indeed, first we note that, by Propositions 31 and 32, we have

$$\begin{aligned} \left| e^{2^{((s-1)n+t)/N}} - \gamma_{s,t} \right| &= \left| e^{2^{((s-1)n+t)/N}} - \sum_{i=0}^k \frac{r_{s,t}^i}{i!} \right| \\ &\leq \left| e^{r_{s,t}} - \sum_{i=0}^k \frac{r_{s,t}^i}{i!} \right| + \left| e^{2^{((s-1)n+t)/N}} - e^{r_{s,t}} \right| \\ &< \frac{1}{k+1} + 42 \left| 2^{((s-1)n+t)/N} - r_{s,t} \right|. \end{aligned}$$

Now using Proposition 30 and recalling the choice of j and k in (24), we derive

$$42 \left| 2^{((s-1)n+t)/N} - r_{s,t} \right| \leq \frac{42}{j+1} \leq \frac{1}{3E} \quad \text{and} \quad \frac{1}{k+1} \leq \frac{1}{3E}$$

and (25) follows.

Thus, by (25) and Proposition 29, computing $\Gamma \mathbf{x}^T$, where

$$\Gamma = (\gamma_{s,t}) \in \mathbb{Q}^{m \times n}$$

requires $m(n-1)$ additions.

Of course, all of the above also holds for computations over \mathbb{C} and unbounded algorithm coefficients, whereas in [7] the algorithm coefficients are bounded by 1.

Even though,

$$\gamma_{s,t} \leq e^{2^{((s-1)n+t)/N}} + \frac{2}{3E} \leq e^2 + 1/6 < 8 \quad (26)$$

the numerator and the denominator of $\gamma_{s,t}$ are very large. The matrix

$$\tilde{\Gamma} = (\tilde{\gamma}_{s,t}) \in \mathbb{Q}^{m \times n} \quad (27)$$

with “smaller” entries such that computing $\tilde{\Gamma} \mathbf{x}^T$ requires $m(n-1)$ additions is defined by

$$\tilde{\gamma}_{s,t} = \frac{\lfloor 3\lfloor E+1 \rfloor \gamma_{s,t} \rfloor}{3\lfloor E+1 \rfloor}, \quad s = 1, \dots, m, \quad t = 1, \dots, n. \quad (28)$$

66:16 Sets of Linear Forms Which Are Hard to Compute

For the proof, by Proposition 29, it suffices to show that

$$\left| e^{2^{((s-1)n+t)/N}} - \tilde{\gamma}_{s,t} \right| < \frac{1}{E},$$

which is indeed so, because

$$\begin{aligned} \left| e^{2^{((s-1)n+t)/N}} - \tilde{\gamma}_{s,t} \right| &= \left| e^{2^{((s-1)n+t)/N}} - \frac{\lfloor 3\lfloor E+1 \rfloor \gamma_{s,t} \rfloor}{3\lfloor E+1 \rfloor} \right| \\ &= \left| e^{2^{((s-1)n+t)/N}} - \gamma_{s,t} - \frac{\{3\lfloor E+1 \rfloor \gamma_{s,t}\}}{3\lfloor E+1 \rfloor} \right| \\ &\leq \left| e^{2^{((s-1)n+t)/N}} - \gamma_{s,t} \right| + \frac{\{3\lfloor E+1 \rfloor \gamma_{s,t}\}}{3\lfloor E+1 \rfloor} \\ &\leq \frac{2}{3E} + \frac{1}{3\lfloor E+1 \rfloor} \leq \frac{1}{E}. \end{aligned}$$

By definition, the matrix

$$\Omega = 3\lfloor E+1 \rfloor \tilde{\Gamma} \in \mathbb{Z}^{m \times n}, \quad (29)$$

where $\tilde{\Gamma}$ is defined by (28), has integer entries and computing $\Omega \mathbf{x}^T$ also requires $m(n-1)$ additions. Thus we have the following.

► **Theorem 33.** *Let $N = mn \geq 4$. The matrix*

$$\Omega = (\omega_{s,t}) \in \mathbb{Z}^{m \times n},$$

given by (29) defines the set of linear forms of complexity $m(n-1)$ and has elements of size

$$0 \leq \omega_{s,t} \leq 25 \exp\left(N^{N^{3N^2}}\right).$$

Proof. From the above discussion it only remains to estimate the size of elements of Ω . From (26) we conclude that

$$0 \leq \omega_{s,t} \leq 8(3E+1) \leq 25E,$$

where E is defined by (23). As one easily verifies that under our assumption we have $E \geq 24$. ◀

It seems to be of interest to find an integer matrix with smaller entries, that defines the set of linear forms of the same complexity.

References

- 1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.
- 2 Tom M. Apostol. *Mathematical Analysis*. Addison-Wesley, Reading, MA, 1963. Second edition.
- 3 Enrico Bombieri and Jeffrey D. Vaaler. On Siegel's lemma. *Inventiones Mathematicae*, 73:11–32, 1983.
- 4 Peter Bürgisser, Michael Clausen, and Amin Shokrollahi. *Algebraic Complexity Theory*. Springer, Berlin, 1997.
- 5 Teresa Krick, Luis M. Pardo, and Martín Sombra. Sharp estimates for the arithmetic Nullstellensatz. *Duke Mathematical Journal*, 9:354–364, 2001.
- 6 Serge Lang. *Algebraic Number Theory*. Addison-Wesley, Reading, MA, 1970.

- 7 Jacques Morgenstern. Note on a lower bound on the linear complexity of the fast Fourier transform. *Journal of the ACM*, 20:305–306, 1973.
- 8 Oskar Perron. *Algebra I (Die Grundlagen)*. Walter de Gruyter, Berlin, 1927.
- 9 Arkadiusz Ploski. Algebraic dependence of polynomials after O. Perron and some applications. In Svetlana Cojocaru, Gerhard Pfister, and Victor Ufnarovski, editors, *Computational Commutative and Non-Commutative Algebraic Geometry*, volume 196 of *NATO Science Series, III: Computer and Systems Sciences*, pages 167–173. IOS Press, Amsterdam, 2005.
- 10 John E. Savage. An algorithm for the computation of linear forms. *SIAM Journal on Computing*, 3:150–158, 1974.
- 11 Alain Sert. Une version effective du théorème de Lindemann-Weierstrass par les déterminants d'interpolation. *Journal of Number Theory*, 76:94–119, 1999.
- 12 Carl L. Siegel. Über einige Anwendungen diophantischer Approximationen. *Abhandlungen der Preussischen Akademie der Wissenschaften. Physikalisch-mathematische Klasse*, 1:209–266, 1929.
- 13 Volker Strassen. Vermeidung von Divisionen. *Journal für Reine und Angewandte Mathematik*, 264:184–202, 1973.
- 14 Volker Strassen. Polynomials with rational coefficients which are hard to compute. *SIAM Journal on Computing*, 3:128–149, 1974.
- 15 Jeffrey D. Vaaler. The best constant in Siegel's lemma. *Monatshefte für Mathematik*, 140:71–89, 2003.
- 16 Shmuel Winograd. On the number of multiplications necessary to compute certain functions. *Communications of Pure and Applied Mathematics*, XXIII:165–179, 1970.

A

 Proofs of Corollary 28 and Proposition 29

A.1 Proof of Corollary 28

We just substitute the upper bounds

$$d \leq N^{N-1} \quad \text{and} \quad h(P) \leq N^{2N^{N^2}}$$

from Corollary 24 on d and H , respectively, in the parameters of Lemma 26 and also recall Remark 27. We note that we use some crude inequalities to simplify the bound.

More precisely, one verifies that for $N \geq 4$ we have

$$41 \times 3^{2N} 2^{-N+1} N^{-1} \leq 2^{5N}.$$

Hence, in Lemma 26 we can now take

$$c_0 \leq 41 \times 3^{2N} 2^{-N+1} D^{N+1} N^{2N-1} \leq 2^{5N} D^{N+1} N^{2N}. \quad (30)$$

Furthermore, simple calculus shows that $\ln(9x) \leq x$ for $x \geq 4$, hence

$$\ln(9DN) \leq DN.$$

Using this and the trivial bounds

$$1 + 6D \leq 2^3 D \quad \text{and} \quad 1 + 6N \leq 2^3 N,$$

66:18 Sets of Linear Forms Which Are Hard to Compute

we now obtain

$$\begin{aligned}
c_1 &= 2^{2-N} 3^{2N+1} D^{N+1} N^N + (1+6D) 2^{4-N} 3^{2N} D^N N^N \ln(9ND) \\
&\quad + 2^{4-N} 3^{2N} D^{N+1} N^N (1+6N) \ln \mathfrak{h}_a(\alpha), \\
&\leq 2^{2-N} 3^{2N+1} D^{N+1} N^N + 2^{7-N} 3^{2N} D^{N+2} N^{N+1} \\
&\quad + 2^{7-N} 3^{2N} D^{N+1} N^{N+1} \ln \mathfrak{h}_a(\alpha), \\
&\leq (2^{2-N} 3^{2N+1} D^{-1} N^{-1} + 2^{7-N} 3^{2N}) D^{N+2} N^{N+1} \\
&\quad + 2^{7-N} 3^{2N} D^{N+1} N^{N+1} \ln \mathfrak{h}_a(\alpha) \\
&\leq (2^{-N} 3^{2N+1} + 2^{7-N} 3^{2N}) D^{N+2} N^{N+1} \\
&\quad + 2^{7-N} 3^{2N} D^{N+1} N^{N+1} \ln \mathfrak{h}_a(\alpha).
\end{aligned}$$

Since, for $N \geq 4$, we have

$$2^{-N} 3^{2N+1} + 2^{7-N} 3^{2N} = (3+128) 2^{-N} 3^{2N} = 131 \times 2^{-N} 3^{2N} \leq 2^{4N}$$

and certainly the same bound for just the second term $2^{7-N} 3^{2N}$, we derive

$$c_1 \leq 2^{4N} D^{N+1} N^{N+1} (D + \ln \mathfrak{h}_a(\alpha)). \quad (31)$$

Finally, for c_2 , using $1+6D \leq 8D$ we have

$$c_2 = 2^{7-N} 3^{2N} D^{N+1} N^N \leq 2^{4N} D^{N+1} N^N. \quad (32)$$

We now collect (30), (31) and (32) and obtain

$$\begin{aligned}
c_0 d^N &\leq 2^{5N} N^{2N} D^{N+1} N^{N(N-1)} = 2^{5N} D^{N+1} N^{N^2+N}, \\
c_1 d^N &\leq 2^{4N} N^{N+1} D^{N+1} (D + \ln \mathfrak{h}_a(\alpha)) N^{N(N-1)} \\
&\quad = 2^{4N} D^{N+2} N^{N^2+1} + 2^{4N} D^{N+1} N^{N^2+1} \ln \mathfrak{h}_a(\alpha), \\
c_2 d^N \ln d &\leq 2^{4N} N^N D^{N+1} N^{N(N-1)} (N-1) \ln N \leq 2^{4N} D^{N+1} N^{N^2+1} \ln N.
\end{aligned}$$

Therefore, we have the inequality

$$\begin{aligned}
c_1 d^N + c_2 d^N \ln d + 72 \max\{1, \mathfrak{h}_a(\alpha)\} \\
\leq 2^{4N} D^{N+2} N^{N^2+1} + 2^{4N} D^{N+1} N^{N^2+1} \ln N \\
+ \left(2^{4N} D^{N+1} N^{N^2+1} + 72 \right) \max\{1, \mathfrak{h}_a(\alpha)\}. \quad (33)
\end{aligned}$$

We now observe that

$$2^{4N} D^{N+2} N^{N^2+1} + 2^{4N} D^{N+1} N^{N^2+1} \ln N \leq 2^{4N+1} D^{N+2} N^{N^2+2} \ln N$$

and

$$2^{4N} D^{N+1} N^{N^2+1} + 72 \leq 2^{4N+1} D^{N+1} N^{N^2+1} \leq 2^{4N+1} D^{N+1} N^{N^2+1} \ln N.$$

Hence, the inequality (33) simplifies as follows:

$$\begin{aligned}
c_1 d^N + c_2 d^N \ln d + 72 \max\{1, \mathfrak{h}_a(\alpha)\} \\
\leq 2^{4N+1} D^{N+1} N^{N^2+2} (D + \max\{1, \mathfrak{h}_a(\alpha)\}) \ln N,
\end{aligned}$$

and thus,

$$\exp(c_1 d^N + c_2 d^N \ln d + 72 \max\{1, \mathfrak{h}_a(\boldsymbol{\alpha})\}) \leq N^{2^{4N+1} D^{N+1} N^{N^2+2} (D + \max\{1, \mathfrak{h}_a(\boldsymbol{\alpha})\})}.$$

Finally

$$\ln H = \ln h(P) \leq 2N^{N^2} \ln N$$

and we conclude

$$\begin{aligned} \ln H + \exp(c_1 d^N + c_2 d^N \ln d + 72 \max\{1, \mathfrak{h}_a(\boldsymbol{\alpha})\}) \\ \leq 2N^{2^{4N+1} D^{N+1} N^{N^2+2} (D + \max\{1, \mathfrak{h}_a(\boldsymbol{\alpha})\})}. \end{aligned}$$

Therefore, combining this bound with the above bound on $c_0 d^N$, by Lemma 26 we have

$$\ln |Q(e^{\alpha_1}, \dots, e^{\alpha_N})| \geq -2^{5N+1} D^{N+1} N^{N^2+N} N^{2^{4N+1} D^{N+1} N^{N^2+2} (D + \max\{1, \mathfrak{h}_a(\boldsymbol{\alpha})\})}.$$

Elementary calculus shows that for $N \geq 4$ we have

$$2^{5N+1} D^{N+1} N^{N^2+N} \leq N^{2^{4N+1} D^{N+1} N^{N^2+1}}.$$

Hence

$$\begin{aligned} \ln |Q(e^{\alpha_1}, \dots, e^{\alpha_N})| &\geq -N^{2^{4N+2} D^{N+1} N^{N^2+2} (D + \max\{1, \mathfrak{h}_a(\boldsymbol{\alpha})\})} \\ &\geq -N^{2^{5N} D^{N+1} N^{N^2+2} (D + \max\{1, \mathfrak{h}_a(\boldsymbol{\alpha})\})}, \end{aligned}$$

and the result follows.

A.2 Proof of Proposition 29

Since $[\mathbb{Q}(\alpha^{1/N}) : \mathbb{Q}] = N$, we see that $\alpha_k = \alpha^{k/N}$, $k = 1, \dots, N$, are linearly independent over \mathbb{Q} and $\mathbb{Q}(\alpha_1, \dots, \alpha_N) = \mathbb{Q}(\alpha_N)$. Thus, for D in Corollary 28 we have $D = N$.

Let $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_N)$. By definition, $h(\boldsymbol{\alpha}) = \alpha$. We contend that $\mathfrak{h}_a(\boldsymbol{\alpha}) = \alpha$ as well.

Since all α_k , $k = 1, \dots, N$, are algebraic integers, their p -adic norms are less than 1. Therefore,

$$\mathfrak{h}_a(\boldsymbol{\alpha}) = \prod_{\nu|\infty} \max_{1 \leq k \leq N} \{1, |\alpha_k|_\nu\}^{D_\nu/N},$$

where (in this context) ∞ denotes the Archimedean value on \mathbb{Q} .

Since for all $\nu | \infty$ and all $k = 1, \dots, N$,

$$\max\{1, |\alpha_k|_\nu\} = \alpha_k = \alpha^{k/N}$$

and by [6, Corollary 1, Section II.1]

$$\sum_{\nu|\infty} D_\nu = N,$$

we derive

$$\mathfrak{h}_a(\boldsymbol{\alpha}) = \prod_{\nu|\infty} \alpha^{D_\nu/N} = \alpha.$$

66:20 Sets of Linear Forms Which Are Hard to Compute

Now, the result follows from Corollary 18 with $\mathfrak{Q} = \mathfrak{Q}_{m,n}$, Corollary 25 and Corollary 28 by simple calculations.

Indeed, let

$$\delta_{s,t} = e^{\alpha^{((s-1)n+t)/N}}, \quad s = 1, \dots, m, \quad t = 1, \dots, n.$$

be the components of the vector $\boldsymbol{\delta} \in \mathbb{R}^N$ (indexed by $(s-1)n+t$).

For any $Q \in \mathfrak{Q}$ and $\boldsymbol{\mu} \in \mathbb{R}^N$ with $|\boldsymbol{\mu}| < 1$, by Corollary 25 (in which we interpret $\boldsymbol{\delta}$ and $\boldsymbol{\mu}$ as N -dimensional vectors), we have

$$|\nabla Q(\boldsymbol{\delta} + \boldsymbol{\mu})| < N^{3N^{N^2}} (e^a + 1)^{N^{N-1}} \leq N^{3N^{N^2}} (e^{N-1} + 1)^{N^{N-1}} \leq N^{3N^{N^2}} e^{N^N} \leq N^{4N^{N^2}}.$$

On the other hand, recalling that $D = N$ and $h(\boldsymbol{\alpha}) = \alpha$, by Corollary 28 we have

$$\begin{aligned} |Q(\boldsymbol{\delta})| &> \exp\left(-N^{2^{5N}} N^{N^2+N+3(N+\alpha)}\right) \\ &\geq \exp\left(-N^{2^{5N}} N^{N^2+N+3(2N-1)}\right) \\ &\geq \exp\left(-N^{2^{5N+1}} N^{N^2+N+4}\right). \end{aligned}$$

Hence

$$\frac{|Q(\boldsymbol{\delta})|}{|\nabla Q(\boldsymbol{\delta} + \boldsymbol{\mu})|} \geq N^{-4N^{N^2}} \exp\left(-N^{2^{5N+1}} N^{N^2+N+4}\right) \geq \exp\left(-2N^{2^{5N+1}} N^{N^2+N+4}\right).$$

Furthermore, since $N \geq 4$, we have

$$2N^{2^{5N+1}} N^{N^2+N+4} \leq N^{2^{5N+2}} N^{N^2+N+4} \leq N^{2^{6N}} N^{N^2+N+4} \leq N^{N^{N^2+4N+4}} \leq N^{N^{3N^2}}.$$

Note that, by (21), the condition $|\Delta - \Gamma| < 1$ of Corollary 18 is redundant.

B Density of matrices defining sets of linear forms of maximal additive complexity

B.1 Complexity of matrices

For a matrix Δ and a vector $\boldsymbol{x} = (x_1, \dots, x_n)^T$ of n indeterminates as in (1) and (2), respectively, we denote the additive complexity of the set of linear forms $\Delta\boldsymbol{x}$ by $\mathcal{C}(\Delta)$ and call it the complexity of Δ .

B.2 The case of finite fields

Let \mathbb{F} be a finite field of q elements. In this section we show that “almost all” $m \times n$ matrices over \mathbb{F} , are of high complexity.

We may assume that $m \leq \frac{q^n - 1}{q - 1}$, because the number of non-zero linear forms in n indeterminates over \mathbb{F} is $q^n - 1$ and each form can be scaled by $q - 1$ non-zero elements of \mathbb{F} .

The proof is by the counting argument similar to that of [10, Lemma, Section 5]. For the sake of completeness, we reproduce it below.

For a positive integers C , m and n we denote by $S(C, m, n)$ the set of all $m \times n$ matrices over \mathbb{F} of complexity not exceeding C :

$$S_{\mathbb{F}}(C, m, n) = \{\Delta \in \mathbb{F}^{m \times n} : \mathcal{C}(\Delta) \leq C\}.$$

First we need a bound on the cardinality $\#S_{\mathbb{F}}(C, m, n)$ of this set.

► **Proposition 34.** *We have*

$$\#S_{\mathbb{F}}(C, m, n) < (C + n)^{2C+m} q^{C+m}.$$

Proof. For a positive integer m a linear algorithm in n indeterminates of length C defines at most

$$L_{\mathbb{F}}(C, m) = \binom{C}{m}$$

sets of m linear forms, each of which is associated with

$$M_{\mathbb{F}}(m) = (q - 1)^m m!$$

matrices of the size $m \times n$. Namely, we have $(q - 1)^m$ scalar multiplications and $m!$ permutations, respectively, of the matrix rows.

Next we are going to count the number of linear algorithms over \mathbb{F} in n indeterminates of length C , denoted by $A_{\mathbb{F}}(C, n)$.

By definition,

- $A_{\mathbb{F}}(1, n) \leq n^2(q - 1)$ and
- $A_{\mathbb{F}}(C + 1, n) \leq A_{\mathbb{F}}(C, n)(C + n)^2(q - 1)$,

where the factors $(C + n)^2$ and $(q - 1)$ come from the last addition $u_{C+1} = v + \alpha w$ of the algorithm,

$$v, w \in \{x_1, \dots, x_n\} \cup \{u_1, \dots, u_C\}$$

and $\alpha \neq 0$. Recall that all algorithms under consideration are normalized.

Solving the above recursion, we obtain

$$A_{\mathbb{F}}(C, n) \leq \left(\frac{(C + n - 1)!}{(n - 1)!} \right)^2 (q - 1)^C.$$

Thus,

$$\begin{aligned} \#S_{\mathbb{F}}(C, m, n) &\leq A_{\mathbb{F}}(C, n) L_{\mathbb{F}}(C, m) M_{\mathbb{F}}(m) \\ &< \left(\frac{(C + n - 1)!}{(n - 1)!} \right)^2 (q - 1)^C \binom{C}{m} (q - 1)^m m! \\ &< (C + n)^{2C+m} (q - 1)^{C+m} < (C + n)^{2C+m} q^{C+m}, \end{aligned}$$

which concludes the proof. ◀

► **Theorem 35.** *Let $\varepsilon, \kappa > 0$ be such that $2\kappa + \varepsilon < 1$ and let*

$$C_{\kappa, m, n} = \frac{\kappa mn}{\log_q(mn)} - n.$$

Then

$$\lim_{\substack{n \rightarrow \infty \\ q^{\varepsilon n} \geq m}} \frac{\#S_{\mathbb{F}}(C_{\kappa, m, n}, m, n)}{q^{mn}} = 0.$$

Proof. It suffices to show that

$$\lim_{\substack{n \rightarrow \infty \\ q^{\varepsilon n} \geq m}} (\log_q \#S_{\mathbb{F}}(C_{\kappa, m, n}, m, n) - mn) = -\infty. \quad (34)$$

66:22 Sets of Linear Forms Which Are Hard to Compute

By Proposition 34 we have

$$\log_q \#S_{\mathbb{F}}(C_{\kappa,m,n}, m, n) - mn < (2C_{\kappa,m,n} + m) \log_q(C_{\kappa,m,n} + n) + C_{\kappa,m,n} + m - mn.$$

Recalling the definition of $C_{\kappa,m,n}$ and that $\kappa \leq 1$, we obtain

$$\begin{aligned} & \log_q \#S_{\mathbb{F}}(C_{\kappa,m,n}, m, n) - mn \\ & < \left(\frac{2\kappa mn}{\log_q(mn)} - 2n + m \right) \log_q \left(\frac{\kappa mn}{\log_q(mn)} \right) + \frac{\kappa mn}{\log_q(mn)} - n + m - mn \\ & \leq \left(\frac{2\kappa mn}{\log_q(mn)} + m \right) \log_q(mn) + \frac{mn}{\log_q(mn)} + m - mn \\ & = 2\kappa mn + m \log_q m + m \log_q n + \frac{mn}{\log_q(mn)} + m - mn. \end{aligned}$$

Under the condition $q^{\varepsilon n} \geq m$, we now obtain

$$\begin{aligned} \log_q \#S_{\mathbb{F}}(C_{\kappa,m,n}, m, n) - mn & < 2\kappa mn + \varepsilon mn + m \log_q n + \frac{mn}{\log_q(mn)} + m - mn \\ & = (2\kappa + \varepsilon - 1 + o(1)) mn, \end{aligned}$$

and since $2\kappa + \varepsilon < 1$, we have (34). ◀

B.3 The case of rationals

In this case matrices with rational entries, since the set of polynomials $\mathfrak{Q}_{m,n}$, defined by (20), is (m, n) -complete, for for each matrix (1) such that $\mathcal{C}(\Delta) < m(n-1)$,

$$Q_{m,n}(\delta_{1,1}, \dots, \delta_{m,n}) = 0,$$

where $Q_{m,n} = \prod_{Q \in \mathfrak{Q}_{m,n}} Q$.

Hence the entries of $m \times n$ rational matrices of complexity $\mathcal{C}(\Delta) < m(n-1)$ form a very sparse set in \mathbb{Q}^{mn} (after we represent them as mn -dimensional vector). Namely, this set is a hypersurface of dimension $mn - 1$. In particular, almost all rational matrices (in terms on natural density) are of the maximal complexity $m(n-1)$.

On Positivity and Minimality for Second-Order Holonomic Sequences

George Kenison ✉

Institute for Logic and Computation,
The Technical University of Vienna, Austria

Engel Lefauchaux ✉

Max Planck Institute for Software Systems,
Saarland Informatics Campus, Saarbrücken,
Germany

Pieter Moree ✉ 

Max Planck Institute for Mathematics,
Bonn, Germany

Markus A. Whiteland ✉ 

Max Planck Institute for Software Systems,
Saarland Informatics Campus, Saarbrücken,
Germany

Oleksiy Klurman ✉

School of Mathematics, University of Bristol, UK
Max Planck Institute for Mathematics,
Bonn, Germany

Florian Luca ✉

School of Mathematics, University of the
Witwatersrand, Johannesburg, South Africa
Research Group in Algebraic Structures &
Applications, King Abdulaziz University, Riyadh,
Saudi Arabia
Centro de Ciencias Matemáticas UNAM,
Morelia, Mexico

Joël Ouaknine ✉ 

Max Planck Institute for Software Systems,
Saarland Informatics Campus, Saarbrücken,
Germany

James Worrell ✉

Department of Computer Science,
University of Oxford, UK

Abstract

An infinite sequence $\langle u_n \rangle_n$ of real numbers is *holonomic* (also known as *P-recursive* or *P-finite*) if it satisfies a linear recurrence relation with polynomial coefficients. Such a sequence is said to be *positive* if each $u_n \geq 0$, and *minimal* if, given any other linearly independent sequence $\langle v_n \rangle_n$ satisfying the same recurrence relation, the ratio $u_n/v_n \rightarrow 0$ as $n \rightarrow \infty$.

In this paper we give a Turing reduction of the problem of deciding positivity of second-order holonomic sequences to that of deciding minimality of such sequences. More specifically, we give a procedure for determining positivity of second-order holonomic sequences that terminates in all but an exceptional number of cases, and we show that in these exceptional cases positivity can be determined using an oracle for deciding minimality.

2012 ACM Subject Classification Theory of computation \rightarrow Logic and verification

Keywords and phrases Holonomic sequences, Minimal solutions, Positivity Problem

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.67

Funding *George Kenison*: WWTF Grant ProbInG ICT19-018 and the ERC Consolidator Grant ARTIST 101002685.

Joël Ouaknine: ERC grant AVS-ISS (648701), and DFG grant 389792660 as part of TRR 248 (see <https://perspicuous-computing.science>).

Joël Ouaknine is also affiliated with Keble College, Oxford as **emmy.network** Fellow.

James Worrell: EPSRC Fellowship EP/N008197/1.

Acknowledgements This work was partly carried out during a visit of Florian Luca at the Max Planck Institute for Software Systems in Saarbrücken, Germany from September 2020 to March 2021. He thanks the institution for its hospitality and excellent working conditions.

The authors would like to thank the anonymous referees for their detailed comments, which have led to significant improvements and clarifications in the final version of this paper.



© George Kenison, Oleksiy Klurman, Engel Lefauchaux, Florian Luca, Pieter Moree, Joël Ouaknine, Markus A. Whiteland, and James Worrell;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 67; pp. 67:1–67:15

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Holonomic sequences (also known as *P-recursive* or *P-finite* sequences) are infinite sequences of real (or complex) numbers that satisfy a linear recurrence relation with polynomial coefficients. Holonomic sequences play a critical role in many areas of mathematics and computer science – particularly combinatorics, analysis of algorithms, and number theory; see, for instance [31, 6, 7] or the seminal paper [40]. A spectacular application can be found in groundbreaking work by Apéry in the 1970s, who used certain holonomic sequences satisfying the second-order recurrence relation

$$n^3 u_n = (34n^3 - 51n^2 + 27n - 5)u_{n-1} - (n-1)^3 u_{n-2}$$

to prove that $\zeta(3) := \sum_{n=1}^{\infty} n^{-3}$ is irrational [1].

Formally, a holonomic sequence satisfies a recurrence relation of the form:

$$p_{k+1}(n)u_n = p_k(n)u_{n-1} + \cdots + p_1(n)u_{n-k}$$

where $p_{k+1}, \dots, p_1 \in \mathbb{Q}[n]$ are polynomials with rational coefficients and $p_1 \neq 0$. We define the *order* of the recurrence to be k . Assuming that $p_{k+1}(n) \neq 0$ for each non-negative integer n , the above recurrence uniquely defines an infinite sequence once the initial values u_{-k+1}, \dots, u_0 are specified. By extension, if a holonomic sequence satisfies a recurrence of order k , but no recurrence of smaller order, then we say that the sequence has order k . The class of holonomic sequences who satisfy recurrence relations with constant (rather than polynomial) coefficients are known as *C-finite* sequences. Furthermore, every algebraic sequence of real numbers (i.e., whose ordinary generating function is algebraic) is also holonomic.

The study of identities for holonomic sequences appears frequently in the literature. However, as noted by Kauers and Pillwein, “*in contrast, . . . almost no algorithms are available for inequalities*” [17]. For example, the *Positivity Problem* (i.e., whether every term of a given sequence is non-negative) for *C-finite* sequences is only known to be decidable at low orders, and there is strong evidence that the problem is mathematically intractable in general [28, 30]; see also [12, 20, 29]. For holonomic sequences that are not *C-finite*, very few decision procedures currently exist for Positivity, although several partial results and heuristics are known (see, for example [17, 21, 26, 27, 32, 33, 39]). In particular, in [27], the authors exhibit semi-decision procedures for determining positivity of second-order holonomic sequences for which the degrees of the polynomial coefficients satisfy certain constraints.

Another extremely important property of holonomic sequences is *minimality*; a sequence $\langle u_n \rangle_n$ is a minimal solution if, given any other linearly independent sequence $\langle v_n \rangle_n$ satisfying the same recurrence relation, the ratio u_n/v_n converges to 0. Minimal holonomic sequences play a crucial rôle, among others, in numerical calculations and asymptotics, as noted for example in [11, 4, 5, 8, 9, 10] – see also the references therein. Unfortunately, there is also ample evidence that determining algorithmically whether a given holonomic sequence is minimal is a very challenging task, for which no satisfactory solution is at present known to exist.

One of our main results concerns the relationship between positivity and minimality of sequences $\langle u_n \rangle_{n=-1}^{\infty}$ satisfying second-order polynomial recurrences:¹

$$p_3(n)u_n = p_2(n)u_{n-1} + p_1(n)u_{n-2}. \tag{1}$$

¹ Indexing the sequence from -1 (rather than the more usual 0) makes no significant mathematical difference, but provides notational expediency in the sequel.

We shall assume throughout that neither p_1 nor p_2 is identically zero. Indeed, if $p_1 \equiv 0$ then $\langle u_n \rangle_{n=-1}^\infty$ satisfies a first-order recurrence, while if $p_2 \equiv 0$ then $\langle u_n \rangle_{n=-1}^\infty$ is the interleaving of two sequences that satisfy first-order recurrences. But it is trivial to determine the positivity of first-order holonomic sequences.² Moreover, by working with a tail of the sequence $\langle u_n \rangle_{n=-1}^\infty$ (equivalently, shifting the index n) we can assume that $p_1(n), p_2(n), p_3(n) \neq 0$ for all $n \geq -1$.

Our main contributions are as follows: we characterise the positivity of the sequence $\langle u_n \rangle_{n=-1}^\infty$ in (1) in terms of its *initial ratio* u_0/u_{-1} . Specifically, from the recurrence we obtain a single closed subinterval $P \subseteq \mathbb{R}$ such that the sequence is positive if and only if $u_0/u_{-1} \in P$. We moreover show that the endpoints of P can be represented as polynomial continued fractions, allowing them to be computed to arbitrary precision. By approximating the endpoints of P to sufficient accuracy we can decide positivity in all cases except when the initial ratio happens to coincide with an endpoint of P . However, we show that such exceptional cases can be handled using an oracle for deciding minimality. Thus we obtain one of our main results, Theorem 3.1: for second-order holonomic sequences, the Positivity Problem Turing-reduces to the Minimality Problem.

2 Preliminaries

2.1 continued fractions

An (infinite) *continued fraction*

$$\mathbf{K}_{n=1}^{\infty} \frac{a_n}{b_n} := \frac{a_1}{b_1 + \frac{a_2}{b_2 + \frac{a_3}{b_3 + \ddots}}}$$

is defined by an ordered pair of sequences $\langle a_n \rangle_n$ and $\langle b_n \rangle_n$ of complex numbers where $a_n \neq 0$ for each $n \in \mathbb{N}$. Herein we shall always assume that $\langle a_n \rangle_n$ and $\langle b_n \rangle_n$ are real-valued rational functions. A continued fraction *converges* to a value $f = \mathbf{K}(a_n/b_n)$ if its *sequence of approximants* $\langle f_n \rangle_{n=1}^\infty$ converges to f in $\hat{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. The sequence $\langle f_n \rangle_n$ is recursively defined so that

$$f_n = \mathbf{K}_{m=1}^n \frac{a_m}{b_m} := \frac{a_1}{b_1 + \frac{a_2}{b_2 + \ddots + \frac{a_n}{b_n}}}$$

We respectively call $\langle a_n \rangle_n$ and $\langle b_n \rangle_n$ the sequences of *partial numerators* and *partial denominators* (together the *partial quotients*) of the continued fraction $\mathbf{K}(a_n/b_n)$. Let $\langle A_n \rangle_{n=-1}^\infty$ and $\langle B_n \rangle_{n=-1}^\infty$ satisfy the recurrence relation $u_n = b_n u_{n-1} + a_n u_{n-2}$ with initial

² The ratio of consecutive terms of a first-order holonomic sequence is a rational function, which has an ultimately constant sign.

values $A_{-1} = 1, A_0 = 0, B_{-1} = 0,$ and $B_0 = 1$. As a pair, $\langle A_n \rangle_{n=-1}^\infty$ and $\langle B_n \rangle_{n=-1}^\infty$ form a basis for the solution space of the recurrence. We call $\langle A_n \rangle_n$ and $\langle B_n \rangle_n$ the sequences of *canonical numerators* and *canonical denominators* of $\mathbb{K}(a_n/b_n)$ because $f_n = A_n/B_n$ for each $n \in \mathbb{N}$.

The following determinant formula is well-known (see, for example, [23, Lemma 4, §IV]).

► **Lemma 2.1.** *Suppose that $\langle u_n \rangle_n$ and $\langle v_n \rangle_n$ are both solutions to the recurrence relation $u_n = b_n u_{n-1} + a_n u_{n-2}$. Then*

$$u_n v_{n-1} - u_{n-1} v_n = (u_0 v_{-1} - u_{-1} v_0) \prod_{k=1}^n (-a_k).$$

Two continued fractions are *equivalent* if they have the same sequence of approximants. The following theorem is attributed to Seidel in [23, §II.2.2].

► **Theorem 2.2.** *The continued fractions $\mathbb{K}(a_n/b_n)$ and $\mathbb{K}(c_n/d_n)$ are equivalent if and only if there exists a sequence $\langle r_n \rangle_{n=0}^\infty$ with $r_0 = 1$ and $r_n \neq 0$ for each $n \in \mathbb{N}$ such that $c_n = r_n r_{n-1} a_n$ and $d_n = r_n b_n$ for each $n \in \mathbb{N}$.*

2.2 Śleszyński–Pringsheim continued fractions

A continued fraction $\mathbb{K}_{n=1}^\infty(a_n/b_n)$ is a *Śleszyński–Pringsheim continued fraction* if $|b_n| \geq |a_n| + 1$ for each $n \in \mathbb{N}$. As before, let $\langle f_n \rangle_n$ be the sequence of approximants associated with such a continued fraction. The following properties are well-known [23, §I.4]. For the open unit interval $(-1, 1) \subset \mathbb{R}$, $a_n/(b_n + (-1, 1)) \subseteq (-1, 1)$ and we have that $f_n \in (-1, 1)$ for each $n \in \mathbb{N}$. Further, it can be shown that $\langle f_n \rangle_n$ converges to a finite value f with $0 < |f| \leq 1$. We will use the following convergence result, which can be derived from the Śleszyński–Pringsheim Theorem (we reproduce the proof in [24, §3.2.4] below).

► **Theorem 2.3.** *Let $\langle f_n \rangle_n$ and $\langle B_n \rangle_n$ be the respective sequences of approximants and canonical denominators for a Śleszyński–Pringsheim continued fraction $\mathbb{K}_{n=1}^\infty(a_n/b_n)$ with $a_n < 0$ and $b_n \geq 1 - a_n$ for each $n \in \mathbb{N}$. Then*

$$B_{n+1} > B_n \geq \sum_{k=0}^n \prod_{m=1}^k (b_m - 1) \geq \sum_{k=0}^n \prod_{m=1}^k |a_m|,$$

$\langle f_n \rangle_n$ is strictly decreasing, and $-1 < f_n < f_{n-1} < 0$.

Proof. We prove by induction that $\langle B_n \rangle_n$ is a strictly increasing sequence. First, $B_0 - B_{-1} = 1$. Second, for our induction hypothesis, let us assume that $B_{n-1} - B_{n-2} > 0$ and $B_{n-2} \geq 0$. Then, using the recurrence relation and our additional assumptions on the coefficients, we have

$$B_n - B_{n-1} = (b_n - 1)B_{n-1} - (-a_n)B_{n-2} \geq (b_n - 1)(B_{n-1} - B_{n-2}).$$

Repeated application of this technique gives

$$B_n - B_{n-1} \geq (B_{n-1} - B_{n-2})(b_n - 1) \geq (B_0 - B_{-1}) \prod_{m=1}^n (b_m - 1) \geq \prod_{m=1}^n |a_m| > 0,$$

from which the desired inequalities follow. We apply the determinant formula to the sequences $\langle A_n \rangle_n$ and $\langle B_n \rangle_n$ (see Lemma 2.1) to obtain

$$f_n - f_{n-1} = -\frac{\prod_{k=1}^n -a_k}{B_n B_{n-1}} < 0.$$

Thus $\langle f_n \rangle_n$ is a strictly decreasing sequence with $f_1 = a_1/b_1 < 0$. The bounds follow from the aforementioned convergence properties of Śleszyński–Pringsheim continued fractions. ◀

2.3 Second-order linear recurrences and continued fractions

Recall that a non-trivial solution $\langle u_n \rangle_{n=-1}^\infty$ of the recurrence $u_n = b_n u_{n-1} + a_n u_{n-2}$ is *minimal* provided that, for all other linearly independent solutions $\langle v_n \rangle_{n=-1}^\infty$ of the same recurrence, we have $\lim_{n \rightarrow \infty} u_n/v_n = 0$. Since the vector space of solutions has dimension two, it is equivalent for a sequence $\langle u_n \rangle_{n=-1}^\infty$ to be minimal for there to exist a linearly independent sequence $\langle v_n \rangle_{n=-1}^\infty$ satisfying the above property. In such cases the solution $\langle v_n \rangle_n$ is called *dominant*.

Note that if $\langle u_n \rangle_n$ and $\langle v_n \rangle_n$ are linearly independent solutions of the above recurrence such that u_n/v_n converges in $\hat{\mathbb{R}}$ then the recurrence relation has a minimal solution [23, §IV]. If, in addition, $\langle u_n \rangle_n$ is minimal then all solutions of the form $\langle c u_n \rangle_n$ where $c \neq 0$ are also minimal. If $\langle u_n \rangle_n$ and $\langle v_n \rangle_n$ are respectively minimal and dominant solutions of the recurrence, then together they form a basis of the solution space.

► **Remark 2.4.** When a second-order recurrence relation admits minimal solutions, it is often beneficial (from a numerical standpoint) to provide a basis of solutions where one of the elements is a minimal solution. Such a basis is used to approximate any element of the vector space of solutions: taking $\langle u_n \rangle_n$ and $\langle v_n \rangle_n$ as above, a general solution $\langle z_n \rangle_n$ is given by $z_n = \alpha_1 u_n + \alpha_2 v_n$.

Let $\langle u_n \rangle_{n=-1}^\infty$ be a non-trivial solution of the recurrence relation $u_n = b_n u_{n-1} + a_n u_{n-2}$, where $a_n \neq 0$ for all n . If $u_{n-1} \neq 0$ then we can rearrange the relation to obtain

$$-\frac{u_{n-1}}{u_{n-2}} = \frac{a_n}{b_n - \frac{u_n}{u_{n-1}}} \quad (2)$$

for each $n \in \mathbb{N}$. In the event that $u_{n-2} = 0$ we take the usual interpretation in $\hat{\mathbb{R}}$. Since $\langle u_n \rangle_n$ is non-trivial and $a_n \neq 0$ for each $n \in \mathbb{N}$, the sequence $\langle u_n \rangle_n$ does not vanish at two consecutive indices. Thus if $u_{n-1} = 0$ then $u_{n-2}, u_n \neq 0$ and so both the left-hand and the right-hand sides of the last equation are well-defined in $\hat{\mathbb{R}}$ and are equal to 0. Thus the sequence with terms $-u_n/u_{n-1}$ is well-defined in $\hat{\mathbb{R}}$ for each $n \in \mathbb{N}$. A sequence $\langle t_n \rangle_{n=0}^\infty$ where $t_n := -u_n/u_{n-1}$ for each $n \in \mathbb{N}$ and $\langle u_n \rangle_n$ non-trivial is called a *tail sequence*. A tail sequence for $\mathbf{K}(a_n/b_n)$ is wholly determined by its initial value t_0 .

Given a convergent continued fraction $\mathbf{K}_{n=1}^\infty(a_n/b_n)$ it is easily shown that the sequence $\langle f^{(m)} \rangle_{m=0}^\infty$ with terms $f^{(m)} := \mathbf{K}_{n=m+1}^\infty(a_n/b_n)$ is a tail sequence. In the literature the sequence $\langle f^{(m)} \rangle_{m=0}^\infty$ is the *sequence of tails* of $\mathbf{K}_{n=1}^\infty(a_n/b_n)$ [23, §2.1].

The next theorem due to Pincherle [34] connects the existence of minimal solutions for a second-order recurrence to the convergence of the associated continued fraction (see also [8, 23, 3]).

► **Theorem 2.5 (Pincherle).** *Let $\langle a_n \rangle_{n=1}^\infty$ and $\langle b_n \rangle_{n=1}^\infty$ be real-valued sequences such that each of the terms a_n is non-zero. First, the recurrence $u_n = b_n u_{n-1} + a_n u_{n-2}$ has a minimal solution if and only if the continued fraction $\mathbf{K}(a_n/b_n)$ converges in $\hat{\mathbb{R}}$. Second, if $\langle u_n \rangle_n$ is a minimal solution of this recurrence then the limit of $\mathbf{K}(a_n/b_n)$ is $-u_0/u_{-1}$. As a consequence, the sequence of canonical denominators $\langle B_n \rangle_{n=-1}^\infty$ is a minimal solution if and only if the value of $\mathbf{K}(a_n/b_n)$ is $\infty \in \hat{\mathbb{R}}$.*

► **Remark 2.6.** The convergence properties of continued fractions whose partial quotients are polynomials has long fascinated researchers. It is notable that the sequence of partial denominators in the continued fraction expansion of $\pi = 3 + \mathbf{K}_{n=1}^{\infty}(1/b_n)$ beginning $\langle b_n \rangle_n = \langle 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, \dots \rangle$ behaves erratically. In contrast, Lord Brouncker (as reported by Wallis in [37]³) gave a continued fraction expansion for $4/\pi$ as follows:

$$\frac{4}{\pi} = 1 + \mathbf{K}_{n=1}^{\infty} \frac{(2n-1)^2}{2}.$$

Likewise, Apéry's constant $\zeta(3)$ has a continued fraction expansion (see [35])

$$\zeta(3) = \frac{6}{5 + \mathbf{K}_{n=1}^{\infty} (-n^6 / (34n^3 + 51n^2 + 27n + 5))}$$

whose partial quotients are ultimately polynomials. Motivated by such constructions, Bowman and Mc Laughlin [2] (see also [25]) coined the term *polynomial continued fraction (PCF)*. A polynomial continued fraction $\mathbf{K}(a_n/b_n)$ has algebraic partial quotients such that for sufficiently large $n \in \mathbb{N}$, a_n and b_n are determined by polynomials in $\mathbb{Q}[n]$.

We call the problem of determining whether a given convergent polynomial continued fraction is equal to a particular algebraic number the *PCF Equality Problem*. The proof of the following corollary is a straightforward application of Theorem 2.5.

► **Corollary 2.7.** *The PCF Equality Problem and the Minimality Problem for second-order holonomic sequences are irreducible.*

Proof. A minimality-preserving transformation takes as input a solution $\langle u_n \rangle_n$ of recurrence $p_3(n)u_n = p_2(n)u_{n-1} + p_1(n)u_{n-2}$ and outputs a solution $\langle v_n \rangle_n$, with $v_n = u_n \prod_{j=0}^n p_3(j)$, of recurrence $v_n = p_2(n)v_{n-1} + p_1(n)p_3(n-1)v_{n-2}$. Clearly, $\langle u_n \rangle_n$ is a minimal solution if and only if $\langle v_n \rangle_n$ is a minimal solution.

The latter of the two recurrence relations is associated with the polynomial continued fraction $\mathbf{K}(a_n/b_n)$ with partial quotients $b_n = p_2(n)$ and $a_n = p_1(n)p_3(n-1)$ for each $n \in \mathbb{N}$. Note that by our assumption that $p_1(n), p_3(n) \neq 0$ for all $n \geq -1$ (see the Introduction) we have that $a_n \neq 0$, as required in our definition of a continued fraction. By Theorem 2.5, $\langle v_n \rangle_n$ is a minimal solution if and only if $\mathbf{K}(a_n/b_n)$ converges to the limit $-v_0/v_{-1}$. Thus if one has an oracle that can determine the value of a polynomial continued fraction, then one can determine whether $\langle v_n \rangle_n$ is a minimal solution. Since minimality is preserved by this transformation, one can determine whether $\langle u_n \rangle_n$ is a minimal solution.

Conversely, given a polynomial continued fraction $\mathbf{K}(a_n/b_n)$ and an algebraic number $\xi \in \mathbb{R}$, let us construct the holonomic sequence $\langle v_n \rangle_{n=-1}^{\infty}$ as follows. For each $n \in \mathbb{N}$, let $v_n = b_n v_{n-1} + a_n v_{n-2}$ with initial conditions $v_{-1} = 1$ and $v_0 = -\xi$. By Theorem 2.5, the sequence $\langle v_n \rangle_n$ is a minimal solution of the recurrence relation if and only if the continued fraction $\mathbf{K}(a_n/b_n)$ converges to the value $-u_0/u_{-1} = \xi$. Hence if one has an oracle that can determine whether a given holonomic sequence is a minimal solution, then one can test the value of a polynomial continued fraction. ◀

Determining whether a given continued fraction converges has attracted much attention (historical accounts are given in [23, 24]). The following theorem collects together results from the literature; the first statement follows as a consequence of Worpitzky's Theorem (see [24, Theorem 3.29]) and the convergence results in [15], whilst the second statement follows from the Lane–Wall characterisation of convergence [24, Theorem 3.3].

³ See the translation by Stedall [38].

► **Theorem 2.8.** *Let $\mathbb{K}(\kappa_n/1)$ be a continued fraction with $\langle \kappa_n \rangle_n$ a function in $\mathbb{Q}(n)$. If $\kappa_n < 0$ for all sufficiently large $n \in \mathbb{N}$, then $\mathbb{K}(\kappa_n/1)$ converges to a value in $\hat{\mathbb{R}}$ if and only if, either*

- $\lim_{n \rightarrow \infty} \kappa_n$ exists and is strictly above $-1/4$, or
- $\lim_{n \rightarrow \infty} \kappa_n = -1/4$ and moreover $\kappa_n \geq -1/4 - 1/(4n)^2 - 1/(4n \log n)^2$ for all sufficiently large n .

► **Remark 2.9.** Note that since κ_n is assumed to be a rational function in the above, the eventual inequality $\kappa_n \geq -1/4 - 1/(4n)^2 - 1/(4n \log n)^2$ can be effectively decided, whence convergence of the continued fraction can be ascertained.

The fact that the coefficients $-1/16$ in Theorem 2.8 are best possible is discussed in [14, 13, 23, 24]. For example, if $\kappa_n = -1/4 - \varepsilon/n^2 + \mathcal{O}(1/n^3)$ where $\varepsilon > 1/16$, or $\kappa_n = -1/4 - \varepsilon_1/n + \mathcal{O}(1/n^2)$ where $\varepsilon_1 > 0$, then the continued fraction $\mathbb{K}(\kappa_n/1)$ diverges. We note that later independent work by Kooman and Tijdeman [19, 18] establishes the same results (as a consequence of their results for linear recurrence sequences).

3 Positivity reduces to Minimality

The goal of this section is to show that, for second-order holonomic sequences, the Positivity Problem Turing-reduces to the Minimality Problem; in other words, given an oracle for the Minimality Problem, one can decide the Positivity Problem.

► **Theorem 3.1.** *For the class of recurrence relations*

$$u_n = b_n u_{n-1} + a_n u_{n-2} \tag{3}$$

whose coefficients are rational functions in $\mathbb{Q}(n)$, the Positivity Problem Turing-reduces to the Minimality Problem.

3.1 reduction argument

Let $\langle u_n \rangle_n$ be a sequence satisfying the second-order relation (1). Recall from the Introduction that we can assume without loss of generality that none of the polynomial coefficients in this recurrence relation has a root $n \geq -1$. Additionally we can assume that $\text{sign}(p_3) = +$ on \mathbb{N} . (Herein we denote the sign of a non-zero number by an element of $\{+, -\}$ with the obvious interpretation.) Thus we define the *signature* of a recurrence relation (1) (or its normalisation (3)) as the ordered pair $(\text{sign}(p_2), \text{sign}(p_1))$. It is useful to consider subcases determined by the signature of the recurrence relation $u_n = b_n u_{n-1} + a_n u_{n-2}$. The Positivity Problem is trivial when the signature of the recurrence is either $(+, +)$ or $(-, -)$. It remains to consider the cases $(-, +)$ and $(+, -)$.

Let $\langle u_n \rangle_n$ satisfy a recurrence with signature $(-, +)$. Then a simple substitution argument gives

$$u_{2n} = (b_{2n} b_{2n-1} + a_{2n} + a_{2n-1} b_{2n}/b_{2n-2}) u_{2n-2} - (a_{2n-1} a_{2n-2} b_{2n}/b_{2n-2}) u_{2n-4}.$$

The sequence of odd terms $\langle u_{2n-1} \rangle_n$ satisfies a similar recurrence relation with signature $(+, -)$. Thus the Positivity Problem for the $(-, +)$ case reduces to determining the Positivity Problem for two recurrences with signature $(+, -)$.

We come to the final case: recurrences with signature $(+, -)$. Let $\langle A_n \rangle_{n=-1}^\infty$ and $\langle B_n \rangle_{n=-1}^\infty$ be the canonical solutions as above. In this case $A_1 = a_1 < 0$ and so one can assume that $u_0 > 0$.⁴ It is useful to normalise recurrence (3). Let $\kappa_n := a_n/(b_n b_{n-1})$, and consider

$$w_n = w_{n-1} + \kappa_n w_{n-2}. \tag{4}$$

Then $\langle w_n \rangle_n$ with $w_{-1} = u_{-1}$ and $w_n := u_n/(\prod_{k=0}^n b_k)$ is a solution to (4) if and only if $\langle u_n \rangle_n$ is a solution to (3). We note minimality, positivity, and signature $(+, -)$ are invariant under this transformation. Such properties follow from our assumption that each $b_n > 0$ and the equivalence transformations for continued fractions in Theorem 2.2.

In light of this reduction, the next result is an immediate corollary of Theorem 2.5 and Remark 2.9.

► **Corollary 3.2.** *Given a recurrence relation of the form (3), it is decidable whether the recurrence admits a minimal solution.*

In the work that follows we split the $(+, -)$ case into subcases depending on whether the limit $\lim_{n \rightarrow \infty} \kappa_n$ exists and, if it exists, its value κ . It turns out that such a recurrence relation admits a non-trivial positive solution if and only if the associated continued fraction converges (see Theorem 2.8).

► **Lemma 3.3.** *Suppose that the continued fraction $\mathbb{K}(\kappa_n/1)$ diverges in $\hat{\mathbb{R}}$. Then there are no non-trivial positive solutions to recurrence (4).*

Proof. Suppose, for a contradiction, that $\langle w_n \rangle_n$ is a positive sequence and non-trivial solution of recurrence (4). Notice that two consecutive terms in $\langle w_n \rangle_n$ cannot both vanish since $\langle w_n \rangle_n$ is non-trivial. Furthermore, $w_n > 0$ for all $n \geq 0$ since otherwise $w_n = 0$ and $w_{n+1} = \kappa_{n+1} w_{n-1} < 0$. We first show that the sequence $\langle B_n \rangle_n$ is also positive.

If $w_{-1} = 0$ then $\langle w_n \rangle_n$ is a constant multiple of $\langle B_n \rangle_n$ and we have nothing to show. Otherwise, $w_{-1} > 0$ and let $\langle \check{w}_n \rangle_n$ be a solution sequence of recurrence (4) such that $\check{w}_{-1} = w_{-1}$ and $\check{w}_0 > w_0$. We then have $\check{w}_n > w_n$ for all $n \in \mathbb{N}$. Indeed, proceeding by induction on n , by Lemma 2.1,

$$\check{w}_n w_{n-1} - \check{w}_{n-1} w_n = (\check{w}_0 w_{-1} - \check{w}_{-1} w_0) \prod_{k=1}^n (-\kappa_k) = (\check{w}_0 - w_0) w_{-1} \prod_{k=1}^n (-\kappa_k) > 0$$

implying that $\check{w}_n w_{n-1} > \check{w}_{n-1} w_n$. The induction hypothesis $\check{w}_{n-1} > w_{n-1}$ (with $n \geq 1$) implies that $\check{w}_{n-1} w_n > w_{n-1} w_n$ as $w_n > 0$ by assumption. It follows that $\check{w}_n w_{n-1} > w_{n-1} w_n$, and thus $\check{w}_n > w_n$.

Notice now that $\langle \check{w}_n - w_n \rangle_n$ is a positive sequence and non-trivial solution of recurrence (4). For each $n \in \{-1, 0, \dots\}$ we have $\check{w}_n - w_n = (\check{w}_0 - w_0) B_n$ and so conclude that $B_n > 0$ for each $n \in \mathbb{N}$.

Let $\langle f_n \rangle_n$ be the sequence of approximants associated with $\mathbb{K}(\kappa_n/1)$. We now apply Lemma 2.1 to the sequences $\langle A_n \rangle_n$ and $\langle B_n \rangle_n$, and our conclusion that $B_n > 0$ for each $n \in \mathbb{N}$, to obtain

$$f_n - f_{n-1} = -\frac{\prod_{k=1}^n -\kappa_k}{B_n B_{n-1}} < 0.$$

Thus $\langle f_n \rangle_n$ is monotonic and therefore convergent in $\hat{\mathbb{R}}$, a contradiction to the divergence of $\mathbb{K}(\kappa_n/1)$. ◀

⁴ Indeed, if $u_0 < 0$ then the sequence is not positive; whereas if $u_0 = 0$, then in turn the sequence is either identically zero, or $u_1 < 0$.

In what follows, “*eventually*” statements shall always assume that a property holds for each $n + N$ where $N \in \mathbb{N}$ is a fixed computable constant. Our assumption on the signature means that $a_{n+N} < 0$ and $b_{n+N} > 0$ for each $n \in \mathbb{N}$. Without loss of generality, we can take $N = 0$ in the upcoming statements and results by considering tails of continued fractions as appropriate.

From Theorem 2.8 and Lemma 3.4 (below) we characterise the boundary for a recurrence relation of the form (4) to admit positive solutions. The proof of Lemma 3.4 uses standard analytic tools for continued fractions of limit parabolic type with a particular choice of parameter sequence $\langle g_n \rangle_n$. More general discussions are given in [16, 22, 23].

► **Lemma 3.4.** *Suppose that eventually*

$$\kappa_n \geq -1/4 - 1/(4n)^2 - 1/(4n \log n)^2. \tag{5}$$

Then the sequence of approximants of the continued fraction $\mathbf{K}_{n=1}^\infty(\kappa_n/1)$ is strictly decreasing and converges to a finite value.

Proof. Without loss of generality we assume that (5) holds for each $n \in \mathbb{N}$. Let $g_0 = g_1 = g_2 = 1$ and $g_n := 1/2 + 1/(4n) + 1/(4n \log n)$ for each $n \geq 3$. The continued fractions $\mathbf{K}_{n=1}^\infty(\kappa_n/1)$ and

$$g_0 \mathbf{K}_{n=1}^\infty \left(\frac{\kappa_n / (g_{n-1} g_n)}{1/g_n} \right) \tag{6}$$

are equivalent; one can prove this assertion by applying Theorem 2.2 with the transformation choice $\tau_n = 1/(b_n g_n)$ for each $n \in \mathbb{N}$. Then, by assumption, $|\kappa_n| \leq g_{n-1}(1 - g_n)$ for each $n \in \mathbb{N}$. Thus

$$1 - \frac{\kappa_n}{g_{n-1} g_n} = \frac{g_{n-1} g_n - \kappa_n}{g_{n-1} g_n} \leq \frac{1}{g_n}.$$

We deduce that the partial numerators and denominators in (6) satisfy the assumptions in Theorem 2.3. Thus the sequence of approximants $\langle f_n \rangle_{n=1}^\infty$ associated with (6) is strictly decreasing and converges to a finite value. The desired result follows. ◀

► **Lemma 3.5.** *Suppose that $\langle w_n \rangle_{n=-1}^\infty$ is a solution to (4) with signature $(+, -)$ such that (5) holds for each $n \in \mathbb{N}$. Let $\langle f_n \rangle_n$ be the sequence of approximants for the associated continued fraction $\mathbf{K}(\kappa_n/1)$. Assume that $w_{-1} > 0$. Given $m \in \mathbb{N}$, we have that $-w_0/w_{-1} < f_m$ if and only if $w_m > 0$.*

Proof. Let $\langle A_n \rangle_n$ and $\langle B_n \rangle_n$ be the sequences of canonical numerators and denominators associated with $\mathbf{K}(\kappa_n/1)$. The continued fractions $\mathbf{K}(\kappa_n/1)$ and (6) are equivalent; in addition, the latter is a Śleszyński–Pringsheim continued fraction whose associated sequence of canonical denominators is non-negative (by Theorem 2.3). The transformation between these two continued fractions preserves the positivity property and so we deduce that each term in $\langle B_n \rangle_n$ is also non-negative.

For each $n \in \mathbb{N}$, $w_n = w_{-1} A_n + w_0 B_n$. Since $B_n > 0$, $-w_0/w_{-1} < A_n/B_n = f_n$ if and only if $w_n > 0$, as desired. ◀

We are now in a position to characterise positive solutions to recurrence (4).

► **Proposition 3.6.** *Suppose that $\langle w_n \rangle_{n=-1}^\infty$ is a solution of recurrence (4) with signature $(+, -)$ such that (5) holds for all $n \in \mathbb{N}$. First, the continued fraction $\mathbf{K}_{n=1}^\infty(\kappa_n/1)$ converges to a finite limit $f < 0$. Second, the sequence $\langle w_n \rangle_{n=-1}^\infty$ with $w_{-1}, w_0 > 0$ is positive if and only if $-w_0/w_{-1} \leq f$.*

Proof. As observed in the proof of Lemma 3.4, $\mathbb{K}(\kappa_n/1)$ and (6) are equivalent continued fractions. The former converges to a negative value $f \in \mathbb{R}$ because the latter is a Śleszyński–Pringsheim continued fraction that satisfies the assumptions in Theorem 2.3.

Let $\langle w_n \rangle_{n=-1}^\infty$ be a solution to recurrence (4). By Lemma 3.5, we have that, for all $n \in \mathbb{N}$, $w_n > 0$ if and only if $-w_0/w_{-1} < f_n$. Moreover, by Theorem 2.3, the sequence $\langle f_n \rangle_n$ is strictly decreasing; it follows that $w_n > 0$ for all $n \in \mathbb{N}$ if and only if $-w_0/w_{-1} \leq f$. ◀

The difficulty one encounters when determining positivity arises when $-w_0/w_{-1}$ is equal to the value f . In other words, we can decide positivity of dominant sequences. Indeed, one can always detect if a non-trivial solution $\langle w_n \rangle_n$ is not positive, i.e., $-w_0/w_{-1} > f$ by computing a sufficient number of terms until one finds an $N \in \mathbb{N}$ such that $w_N < 0$. The dominant positive sequences are considered in the following proposition whose proof is delayed to Section 4.

► **Proposition 3.7.** *Let $\langle w_n \rangle_{n=-1}^\infty$ be a non-trivial solution of (4) with signature $(+, -)$ and suppose that (5) holds for each $n \in \mathbb{N}$. Then one can detect if $-w_0/w_{-1} < f$.*

We deduce that if one can decide whether a holonomic sequence $\langle u_n \rangle_n$ that solves recurrence (3) is minimal, then one can decide whether $\langle u_n \rangle_n$ is a positive solution.

Proof of Theorem 3.1. Assume that one has an oracle for the Minimality Problem for solutions $\langle u_n \rangle_{n=-1}^\infty$ to recurrences of the form (3). Note that if $\langle u_n \rangle_{n=-1}^\infty$ is *not* positive, this can be substantiated in finite time by simple enumeration. It thus remain to show how one can ascertain positivity. We can assume without loss of generality that the recurrence has signature $(+, -)$. As previously mentioned, the problem of determining the positivity of solutions $\langle u_n \rangle_n$ of (3) is equivalent to the problem of determining the positivity of solutions $\langle w_n \rangle_n$ of (4).

Consider a recurrence relation of the form (4) with signature $(+, -)$. By Theorem 2.8, we can decide whether or not $\mathbb{K}(\kappa_n/1)$ converges. If $\mathbb{K}(\kappa_n/1)$ diverges, then by Lemma 3.3, the recurrence has no non-trivial positive solutions. Suppose now that $\mathbb{K}(\kappa_n/1)$ converges to $f \in \hat{\mathbb{R}}$. Then, by Remark 2.9, inequality (5) holds; by Proposition 3.6, it follows that f is finite and a given solution $\langle w_n \rangle_n$ is positive if and only if $-w_0/w_{-1} \leq f$. The condition $-w_0/w_{-1} < f$ is recursively enumerable by Proposition 3.7. Finally, by Theorem 2.5, $-w_0/w_{-1} = f$ if and only if the sequence is minimal, and hence equality of $-w_0/w_{-1}$ and f can be checked by an oracle for Minimality. ◀

3.2 A characterisation of positivity

We end this section by characterising positive solutions to recurrence (3) in terms of the ratio of the initial terms belonging to a certain closed interval. Here we understand a closed interval to be empty, a single point, an interval with finite endpoints, or a half-line (including ∞).

► **Proposition 3.8.** *Consider a recurrence of the form (3) for which the coefficients $\langle a_n \rangle_n$ and $\langle b_n \rangle_n$ have constant sign on \mathbb{N} . There exists a closed interval P such that a non-trivial solution $\langle u_n \rangle_n$, with $u_{-1}, u_0 \geq 0$, to the recurrence is positive if and only if $u_0/u_{-1} \in P$. Moreover, the endpoints of the interval can be expressed using polynomial continued fractions.*

Proof. In the cases where the recurrence has signature $(+, +)$ or $(-, -)$, we can set $P = [0, \infty) \cup \{\infty\}$ ($[0, \infty]$ for short) and $P = \emptyset$, respectively.

Consider now a recurrence with signature $(+, -)$. If it is of the form (4), then we have the following: if the associated continued fraction $\mathbf{K}(\kappa_n/1)$ diverges, then there are no non-trivial positive solutions by Lemma 3.3, and we set $P = \emptyset$. If it converges to f (in particular (5) holds), we may set $P = [-f, \infty]$ as is immediate from Proposition 3.6.

Assume then that the recurrence is not of the form (4). The transformation from $\langle u_n \rangle_n$ to a solution $\langle w_n \rangle_n$ to a recurrence of the form (4) preserves minimality and positivity. Hence, if the associated continued fraction $\mathbf{K}(a_n/b_n)$ does not converge, then there are no non-trivial positive solutions and we may set $P = \emptyset$. If it converges, then so does the continued fraction $\mathbf{K}(\kappa_n/1)$; say it converges to f . Then a non-trivial solution to the recurrence is positive if and only if $u_0/u_{-1} = w_0/(b_0w_{-1}) \in [-f/b_0, \infty]$.

We are left with recurrences (3) with signature $(-, +)$. Let $\langle f_n \rangle_n$ denote the sequence of approximants of the associated continued fraction. One can show by straightforward induction that the sequence $\langle B_n \rangle_n$ is alternating in sign for $n \in \mathbb{N}$: the even terms are positive and the odd terms are negative. Let us write a solution $\langle u_n \rangle_n$, with $u_{-1}, u_0 \geq 0$, as $u_n = u_{-1}A_n + u_0B_n$. We have $u_n \geq 0$ if and only if $u_0/u_{-1} \geq -A_n/B_n = -f_n$ when n is even and $u_0/u_{-1} \leq -A_n/B_n = -f_n$ when n is odd. Now the continued fraction $\mathbf{K}(a_n/b_n)$ is equivalent to $-\mathbf{K}(a_n/-b_n)$. By [23, Theorem 2, §III], $\langle -f_{2n} \rangle_n$ is strictly increasing and has finite limit $-f'$, while $\langle -f_{2n-1} \rangle_n$ is strictly decreasing and has finite limit $-f''$. Moreover, $-f' \leq -f''$. It follows that $\langle u_n \rangle_n$ is positive if and only if $u_0/u_{-1} \in [-f', -f'']$.

That the (finite) endpoints of the above intervals can be described using polynomial continued fractions follows from similar minimality-preserving transformations as performed in the proof of Corollary 2.7 and, in the case of the points f', f'' , from results in [23, §II.2.4]. \blacktriangleleft

4 Detecting positive and dominant solutions

The goal of this section is to prove Proposition 3.7, as such, we will suppose that (5) holds for each $n \in \mathbb{N}$ in the following. The proof follows from the results in Corollary 4.2 and Corollary 4.6.

Broadly speaking, we describe a semi-algorithm with inputs $\langle w_n \rangle_n$. This semi-algorithm terminates in finite time for sequences that are dominant with output “*input is a positive sequence*” or “*input is not a positive sequence,*” as appropriate. The semi-algorithm is non-terminating when given a minimal solution as an input. In terminating instances, the running time depends upon the distance between $-w_0/w_{-1}$ and $\mathbf{K}_{n=1}^\infty(\kappa_n/1)$.

The sequence of approximants $\langle f_n \rangle_{n=1}^\infty$ associated with $\mathbf{K}_{n=1}^\infty(\kappa_n/1)$ is recursively defined by a composition of linear fractional transformations $f_n := s_1 \circ \dots \circ s_n(0)$ where $s_n(w) = \kappa_n/(1+w)$ for each $n \in \{1, 2, \dots\}$ and $w \in \hat{\mathbb{R}}$. The tail sequences of $\mathbf{K}(\kappa_n/1)$ are also recursively defined by linear fractional transformations: given such a tail sequence $\langle t_n \rangle_n$, $s_n^{-1}(t_{n-1}) = t_n$ for each $n \in \mathbb{N}$ (by (2)).

For each n , the linear fractional transformation s_n above has two fixed points $\omega_n^\pm := \frac{1}{2}(-1 \pm \sqrt{1 + 4\kappa_n})$. By (5), $\langle \sqrt{1 + 4\kappa_n} \rangle_n$ converges to a real value. We split our analysis into two cases depending on whether κ_n converges to $-1/4$. These subcases are common in the literature (cf. [23, §5]) as some of the convergence properties of the continued fraction $\mathbf{K}(\kappa/1)$ (with $\kappa := \lim_{n \rightarrow \infty} \kappa_n$) hold for the continued fraction $\mathbf{K}(\kappa_n/1)$. In fact, the subcases of *limit hyperbolic-* and *parabolic type* are named for the classification of the limiting linear fractional transformation $s(w) = \kappa/(1+w)$.

4.1 limit hyperbolic type

A continued fraction $\mathbb{K}(\kappa_n/1)$ is of *limit hyperbolic type* if the finite value $\kappa := \lim_{n \rightarrow \infty} \kappa_n$ satisfies $\kappa > -1/4$. In this case the sequences $\langle \omega_n^+ \rangle_n$ and $\langle \omega_n^- \rangle_n$ converge to distinct limits ω^+ and ω^- , respectively. We shall assume, without loss of generality, that aforementioned eventually statements hold for each $n \in \mathbb{N}$.

The next result is given in the literature. A more general result for asymptotic properties of tail sequences associated with a continued fraction of limit hyperbolic type is given in [24, Theorem 4.13].

► **Theorem 4.1.** *Suppose that $\mathbb{K}(\kappa_n/1)$ is of limit hyperbolic type. The sequence of tails $\langle f^{(n)} \rangle_n$ converges to ω^+ . A tail sequence $\langle t_n \rangle_n$ with $t_0 \neq f^{(0)}$ converges to ω^- .*

► **Corollary 4.2.** *Suppose that $\mathbb{K}(\kappa_n/1)$ is of limit hyperbolic type. One can detect if a solution sequence $\langle w_n \rangle_n$ of recurrence (4) is positive and dominant.*

Proof. Let $\langle t_n \rangle_n$ be the tail sequence associated with a non-trivial solution $\langle w_n \rangle_n$. By Theorem 4.1, a tail sequence $\langle t_n \rangle_n$ associated with a dominant solution converges to ω^- in the limit, whilst the tail sequence $\langle f^{(n)} \rangle_n$ associated with a minimal solution converges to ω^+ in the limit.

There is a computable $N \in \mathbb{N}$ such that for all $n \geq N$, the two fixed points of s_n^{-1} are separated: $\omega_n^- < (\omega^- + \omega^+)/2 = -1/2 < \omega_n^+$. If $m > N$ and $t_m < -1/2$ then $\langle t_{n+m} \rangle_n$ is bounded from above by $-1/2$. This is established by induction. The base case is ensured by the assumption $t_m < -1/2$. Now let $n \geq m$ such that $t_n < -1/2$, we have that $t_{n+1} = s_{n+1}^{-1}(t_n) = \frac{\kappa_{n+1}}{t_n} - 1$. Assume first that $\omega_{n+1}^- < t_n$, then $t_{n+1} = s_{n+1}^{-1}(t_n) \leq t_n < -1/2$ as t_n lies between the two fixed points of s_{n+1} . Otherwise, if $\omega_{n+1}^- \geq t_n$, then

$$t_{n+1} = s_{n+1}^{-1}(t_n) = \frac{\kappa_{n+1}}{t_n} - 1 \leq \frac{\kappa_{n+1}}{\omega_{n+1}^-} - 1 = s_{n+1}^{-1}(\omega_{n+1}^-) = \omega_{n+1}^- < -1/2$$

which completes the induction step.

Thus we can detect if a tail sequence is associated with a dominant solution. Moreover, this observation allows us to detect whether a dominant solution is positive in finite time. ◀

4.2 limit parabolic type

A continued fraction $\mathbb{K}(\kappa_n/1)$ is of *limit parabolic type* if $\lim_{n \rightarrow \infty} \kappa_n = -1/4$. In this case both $\langle \omega_n^+ \rangle_n$ and $\langle \omega_n^- \rangle_n$ converge to $-1/2$.

The limit parabolic case is subtler than the limit hyperbolic case; this is best illustrated by the following result: all tail sequences converge to the same limit (see the general case [24, Theorem 4.17]).

► **Theorem 4.3.** *Let $\mathbb{K}(\kappa_n/1)$ be a continued fraction of limit parabolic type such that (5) holds for each $n \in \mathbb{N}$. Each tail sequence $\langle t_n \rangle_n$ associated with $\mathbb{K}(\kappa_n/1)$ converges to $-1/2$.*

From our assumption that (5) holds for each $n \in \mathbb{N}$, we have bounds on the sequence of tails $f^{(n-1)} := \mathbb{K}_{m=n}^{\infty}(\kappa_m/1)$ by the following generalisation of Worpitzky's Theorem (see, for example, [24, Theorem 3.30]).

► **Theorem 4.4.** *Let $\langle \kappa_n \rangle_n$ be a sequence such that (5) holds for $n \in \mathbb{N}$. Then $\mathbb{K}(\kappa_n/1)$ converges to a finite value f with $0 < |f| \leq 1$ and $|f^{(n)}| \leq g_n$ for each n where $g_0 = 1$ and $g_n := 1/2 + 1/(4n) + 1/(4n \log n)$ for each $n \in \mathbb{N}$.*

The inequalities given in the proof of the next lemma follow from the observation that $s_n^{-1}: (-\infty, 0) \rightarrow (-1, \infty)$ given by $s_n^{-1}(w) = -1 + \kappa_n/w$ is a monotonic bijection.

► **Lemma 4.5.** *Suppose that $\mathbb{K}(\kappa_n/1)$ is of limit parabolic type such that (5) holds for each $n \in \mathbb{N}$. Let $\langle t_n \rangle_n$ be a tail sequence such that $f^{(0)} - t_0 > 0$. Then there exists an $N \in \mathbb{N}$ such that $t_N < -g_N < f^{(N)}$.*

Proof. First, note that one can deduce from Theorem 4.4 that $-g_n < f^{(n)} < 0$ for each $n \in \mathbb{N}$ (otherwise there is an m such that $f^{(m+1)} < -g_{m+1}$). Now let $\langle t_n \rangle_n$ be a tail sequence associated with the continued fraction $\mathbb{K}_{n=1}^{\infty}(\kappa_n/1)$ such that $f^{(0)} - t_0 > 0$. Suppose, for a contradiction, that $-g_n < t_n$ for each $n \in \mathbb{N}$. Thus $\sum_{n=1}^{\infty} \prod_{k=1}^n (-\frac{1+t_n}{t_n})$ diverges to ∞ by comparison with $\sum_{n=1}^{\infty} \prod_{k=1}^n (\frac{1-g_n}{g_n})$; the latter series is known to diverge as $\mathbb{K}_{n=1}^{\infty}(\frac{-g_{n-1}(1-g_n)}{1-g_n})$ is a convergent continued fraction (the full argument, which is beyond the scope of this paper, is given in [36]). However, by Waadeland's Tail Theorem [36, Theorem 1], divergence of $\sum_{n=1}^{\infty} \prod_{k=1}^n (-\frac{1+t_n}{t_n})$ implies that $t_0 = \mathbb{K}_{n=1}^{\infty}(\kappa_n/1) = f^{(0)}$, which contradicts our assumption that $f^{(0)} - t_0 > 0$. ◀

► **Corollary 4.6.** *Suppose that $\mathbb{K}(\kappa_n/1)$ is of limit parabolic type such that (5) holds for each $n \in \mathbb{N}$. One can detect if a solution sequence $\langle w_n \rangle_n$ of recurrence (4) is positive and dominant.*

Proof. Let $\langle t_n \rangle_n$ be the tail sequence associated with a non-trivial solution $\langle w_n \rangle_n$. If $\langle w_n \rangle_n$ is dominant and positive, one has $f^{(0)} - t_0 > 0$ by Proposition 3.6. Moreover, by Lemma 4.5, there exists an $N \in \mathbb{N}$ such that for $t_N < -g_N < f^{(N)}$. Hence one can use the threshold of $-g_N$ to detect whether a solution sequence is dominant and positive. ◀

References

- 1 Roger Apéry. Irrationalité de $\zeta(2)$ et $\zeta(3)$. *Astérisque*, 61:11–13, 1979. Luminy Conference on Arithmetic.
- 2 Douglas Bowman and James Mc Laughlin. Polynomial continued fractions. *Acta Arith.*, 103(4):329–342, 2002.
- 3 Annie Cuyt, Vigdis B. Petersen, Brigitte Verdonk, Haakon Waadeland, and William B. Jones. *Handbook of continued fractions for special functions*. Springer, New York, 2008.
- 4 Alfred Deaño and Javier Segura. Transitory minimal solutions of hypergeometric recursions and pseudoconvergence of associated continued fractions. *Mathematics of Computation*, 76(258):879–901, 2007.
- 5 Alfred Deaño, Javier Segura, and Nico M. Temme. Computational properties of three-term recurrence relations for Kummer functions. *J. Computational Applied Mathematics*, 233(6):1505–1510, 2010.
- 6 Graham Everest, Alfred J. van der Poorten, Igor E. Shparlinski, and Thomas Ward. *Recurrence Sequences*, volume 104 of *Mathematical surveys and monographs*. American Mathematical Society, 2003.
- 7 Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.
- 8 Walter Gautschi. Computational aspects of three-term recurrence relations. *SIAM Rev.*, 9:24–82, 1967.
- 9 Walter Gautschi. Anomalous Convergence of a Continued Fraction for Ratios of Kummer Functions. *Mathematics of Computation*, 31(140):994–999, 1977.
- 10 Walter Gautschi. Minimal solutions of three-term recurrence relations and orthogonal polynomials. *Mathematics of Computation*, 36(154), 1981.

- 11 Amparo Gil, Javier Segura, and Nico M. Temme. *Numerical Methods for Special Functions*, chapter 4, pages 87–122. SIAM, 2007.
- 12 Vesa Halava, Tero Harju, and Mika Hirvensalo. Positivity of second order linear recurrent sequences. *Discrete Appl. Math.*, 154(3):447–451, 2006.
- 13 Lisa Jacobsen. On the convergence of limit periodic continued fractions $K(a_n/1)$, where $a_n \rightarrow -\frac{1}{4}$. II. In *Analytic theory of continued fractions, II (Pitlochry/Aviemore, 1985)*, volume 1199 of *Lecture Notes in Math.*, pages 48–58. Springer, Berlin, 1986.
- 14 Lisa Jacobsen and Alphonse Magnus. On the convergence of limit periodic continued fractions $K(a_n/1)$, where $a_n \rightarrow -\frac{1}{4}$. In Peter Russell Graves-Morris, Edward B. Saff, and Richard S. Varga, editors, *Rational Approximation and Interpolation*, pages 243–248, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg.
- 15 Lisa Jacobsen and David R. Masson. On the convergence of limit periodic continued fractions $K(a_n/1)$, where $a_n \rightarrow -\frac{1}{4}$. III. *Constr. Approx.*, 6(4):363–374, 1990.
- 16 Lisa Jacobsen and David R. Masson. A sequence of best parabola theorems for continued fractions. *Rocky Mountain J. Math.*, 21(1):377–385, March 1991.
- 17 Manuel Kauers and Veronika Pillwein. When can we detect that a P-finite sequence is positive? In Wolfram Koepf, editor, *Symbolic and Algebraic Computation, International Symposium, ISSAC 2010, Munich, Germany, July 25-28, 2010, Proceedings*, pages 195–201. ACM, 2010.
- 18 Robert-Jan Kooman. *Convergence properties of recurrence sequences*. Centrum voor Wiskunde en Informatica, 1991.
- 19 Robert-Jan Kooman and Robert Tijdeman. Convergence properties of linear recurrence sequences. *Nieuw Arch. Wisk. (4)*, 8(1):13–25, 1990.
- 20 Vichian Laohakosol and Pinthira Tangsupphathawat. Positivity of third order linear recurrence sequences. *Discrete Appl. Math.*, 157(15):3239–3248, 2009.
- 21 Lily L. Liu. Positivity of three-term recurrence sequences. *Electron. J. Combin.*, 17(1):Research Paper 57, 10, 2010.
- 22 Lisa Lorentzen. Computation of limit periodic continued fractions. A survey. *Numerical Algorithms*, 10(1):69–111, 1995.
- 23 Lisa Lorentzen and Haakon Waadeland. *Continued fractions with applications*, volume 3 of *Studies in Computational Mathematics*. North-Holland Publishing Co., Amsterdam, 1992.
- 24 Lisa Lorentzen and Haakon Waadeland. *Continued fractions. Vol. 1*, volume 1 of *Atlantis Studies in Mathematics for Engineering and Science*. Atlantis Press, Paris; World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, second edition, 2008.
- 25 James Mc Laughlin and N. J. Wyshinski. Real numbers with polynomial continued fraction expansions. *Acta Arith.*, 116(1):63–79, 2005.
- 26 Marc Mezzarobba and Bruno Salvy. Effective bounds for P-recursive sequences. *J. Symbolic Comput.*, 45(10):1075–1096, 2010.
- 27 Eike Neumann, Joël Ouaknine, and James Worrell. Decision problems for second-order holonomic recurrences. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPICs*, pages 99:1–99:20, 2021.
- 28 Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 366–379. ACM, New York, 2014.
- 29 Joël Ouaknine and James Worrell. Ultimate positivity is decidable for simple linear recurrence sequences. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8–11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 330–341. Springer, 2014.
- 30 Joël Ouaknine and James Worrell. On linear recurrence sequences and loop termination. *SIGLOG News*, 2(2):4–13, 2015.
- 31 Marko Petkovišek, Herbert Wilf, and Doron Zeilberger. *A=B*. A. K. Peters, 1997.

- 32 Veronika Pillwein. Termination conditions for positivity proving procedures. In Manuel Kauers, editor, *International Symposium on Symbolic and Algebraic Computation, ISSAC'13, Boston, MA, USA, June 26–29, 2013*, pages 315–322. ACM, 2013.
- 33 Veronika Pillwein and Miriam Schussler. An efficient procedure deciding positivity for a class of holonomic functions. *ACM Comm. Computer Algebra*, 49(3):90–93, 2015.
- 34 Salvatore Pincherle. Delle Funzioni ipergeometriche, e di varie questioni ad esse attinenti. *Giorn. Mat. Battaglini*, 32:209–291, 1894.
- 35 Alfred van der Poorten. A proof that Euler missed. . . Apéry’s proof of the irrationality of $\zeta(3)$. *Math. Intelligencer*, 1(4):195–203, 1979.
- 36 Haakon Waadeland. Tales about tails. *Proceedings of the American Mathematical Society*, 90(1):57–57, 1984.
- 37 John Wallis. *Arithmetica infinitorum, sive nova methodus inquirendi in curvilinearum quadraturam, aliaque difficiliori matheseos problemata*. Oxford, pages 1–199, 1655.
- 38 John Wallis. *The arithmetic of infinitesimals*. Sources and Studies in the History of Mathematics and Physical Sciences. Springer-Verlag, New York, 2004. Translated from the Latin and with an introduction by Jacqueline A. Stedall.
- 39 Ernest X. W. Xia and X. M. Yao. The signs of three-term recurrence sequences. *Discrete Applied Mathematics*, 159(18):2290–2296, 2011.
- 40 Doron Zeilberger. A holonomic systems approach to special functions identities. *Journal of Computational and Applied Mathematics*, 32(3):321–368, 1990.

Improved Upper Bounds for the Rigidity of Kronecker Products

Bohdan Kivva   

University of Chicago, IL, USA

Abstract

The *rigidity* of a matrix A for target rank r is the minimum number of entries of A that need to be changed in order to obtain a matrix of rank at most r . At MFCS'77, Valiant introduced matrix rigidity as a tool to prove circuit lower bounds for linear functions and since then this notion received much attention and found applications in other areas of complexity theory. The problem of constructing an explicit family of matrices that are sufficiently rigid for Valiant's reduction (Valiant-rigid) still remains open. Moreover, since 2017 most of the long-studied candidates have been shown not to be Valiant-rigid.

Some of those former candidates for rigidity are Kronecker products of small matrices. In a recent paper (STOC'21), Alman gave a general non-rigidity result for such matrices: he showed that if an $n \times n$ matrix A (over any field) is a Kronecker product of $d \times d$ matrices M_1, \dots, M_k (so $n = d^k$) ($d \geq 2$) then changing only $n^{1+\varepsilon}$ entries of A one can reduce its rank to $\leq n^{1-\gamma}$, where $1/\gamma$ is roughly $2^d/\varepsilon^2$.

In this note we improve this result in two directions. First, we do not require the matrices M_i to have equal size. Second, we reduce $1/\gamma$ from exponential in d to roughly $d^{3/2}/\varepsilon^2$ (where d is the maximum size of the matrices M_i), and to nearly linear (roughly d/ε^2) for matrices M_i of sizes within a constant factor of each other.

As an application of our results we significantly expand the class of Hadamard matrices that are known not to be Valiant-rigid; these now include the Kronecker products of Paley-Hadamard matrices and Hadamard matrices of bounded size.

2012 ACM Subject Classification Theory of computation; Theory of computation \rightarrow Complexity theory and logic

Keywords and phrases Matrix rigidity, Kronecker product, Hadamard matrices

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.68

Related Version *Previous Version:* <https://arxiv.org/abs/2103.05631>

Funding PhD student, partially supported by Prof. László Babai's NSF Grant CCF 1718902. All statements made in this paper reflect the author's views and have not been evaluated or endorsed by the NSF.

Acknowledgements The author is grateful to his advisor László Babai for helpful discussions, his help in improving the organization of the paper, and for pointing out improvements to the results and simplifications of the proofs.

1 Introduction

1.1 Recent upper bounds on rigidity

In his celebrated MFCS'77 paper [13], Leslie Valiant introduced the notion of matrix rigidity as a tool to prove lower bounds for arithmetic circuits. Since then, several other important problems in complexity theory have been reduced to proving rigidity lower bounds for explicit families of matrices (see, e.g., [12, 7] and the survey [10]).



© Bohdan Kivva;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 68; pp. 68:1–68:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

► **Definition 1.** Let \mathbb{F} be a field. For a matrix $A \in \mathbb{F}^{n \times m}$ and a target rank $0 \leq r \leq \min(n, m)$ let $R_{\mathbb{F}}(A, r)$ denote the minimum number of non-zero entries in a matrix $Z \in \mathbb{F}^{n \times m}$ such that $\text{rank}(A - Z) \leq r$. The function $R_{\mathbb{F}}(A, \cdot)$ is called the rigidity of A over \mathbb{F} .

Valiant [13] proved that if for some $\varepsilon > 0$ the sequence of matrices $A_n \in \mathbb{F}^{n \times n}$ satisfies

$$R_{\mathbb{F}}(A_n, n/\log \log n) \geq n^{1+\varepsilon}, \quad (1)$$

then the linear functions $x \mapsto A_n x$ cannot be computed by arithmetic circuits of size $O(n)$ and depth $O(\log n)$. Following [5], we say that a family of matrices A_n is *Valiant-rigid* if it satisfies Eq. (1) for some $\varepsilon > 0$ and all sufficiently large n . By saying that a family \mathcal{F} of matrices is not Valiant-rigid we mean that none of the subsequences of matrices of \mathcal{F} of increasing order is Valiant-rigid.

The problem of constructing explicit Valiant-rigid matrices has been attacked for more than four decades, but still remains open (see the survey [10]). Over this time a few candidates of Valiant-rigid families were proposed that included Hadamard matrices [11, 12], circulants [4], Discrete Fourier Transform (DFT) matrices [13], incidence matrices of projective planes over a finite field [13].

In 2017, Alman and Williams [2] proved, that, contrary to expectations, the Walsh–Hadamard matrices are not Valiant-rigid over \mathbb{Q} . Subsequently, most of other long-studied candidates for rigidity were shown not to be Valiant-rigid. Dvir and Edelman [5] proved that G -circulants¹ are not Valiant-rigid over \mathbb{F}_p for G the additive group of \mathbb{F}_p^n . Dvir and Liu [6] proved that DFT matrices, circulant matrices, and more generally, G -circulant matrices for any abelian group G , are not Valiant-rigid over \mathbb{C} . Moreover, as observed in [3], the results of Dvir and Liu imply that the Paley–Hadamard matrices and the Vandemonde matrices with a geometric progression as generators are not Valiant-rigid over \mathbb{C} , and the incidence matrices of projective planes over finite fields are not Valiant-rigid either over \mathbb{F}_2 (contrary to Valiant’s suggestion [13]) or over \mathbb{C} .

Upper bounds on the rigidity of the Kronecker powers of a fixed matrix play an important role in these results. Indeed, the Walsh–Hadamard matrices are just the Kronecker powers of $H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. Furthermore, in order to show that DFT and circulant matrices are not Valiant-rigid, as the first step, Dvir and Liu [6] prove that generalized Walsh–Hadamard matrices (Kronecker powers of a DFT matrix) are not Valiant-rigid.

Hence, one may expect that strong upper bounds on the rigidity of Kronecker products of small matrices may lead to upper bounds on the rigidity of interesting new families of matrices, and so will further contribute to our intuition of where not to look for Valiant-rigid matrices.

Recently, Josh Alman [1] proved that Kronecker products of square matrices of any *fixed* size are not Valiant-rigid. More precisely, he proved the following result.

► **Theorem 2 (Alman).** Given $d \geq 2$ and $\varepsilon > 0$, there exists $\gamma = \Omega\left(\frac{d \log d}{2^d} \cdot \frac{\varepsilon^2}{\log^2(1/\varepsilon)}\right)$ such that the following holds for any sequence of matrices $M_1, M_2, \dots, M_k \in \mathbb{F}^{d \times d}$. Let $M = \bigotimes_{i=1}^k M_i$ and $n = d^k$. Then $R_{\mathbb{F}}(M, n^{1-\gamma}) \leq n^{1+\varepsilon}$.

¹ For a finite abelian group G , a G -circulant is a $|G| \times |G|$ matrix of the form $M(f)$ with entries $M(f)_{xy} = f(x - y)$ ($x, y \in G$) where f is any function with domain G .

1.2 Our results: improved bounds, non-uniform sizes

In this note we improve Alman’s result in two directions. First, we do not require the matrices M_i to have equal size. Second, we reduce $1/\gamma$ from exponential in d to roughly $d^{3/2}/\varepsilon^2$ (where d is the maximum size of the matrices), and to nearly linear (roughly d/ε) for matrices M_i of sizes within a constant factor of each other.

This means that for matrices of equal size we get a meaningful reduction of the rank already for $k = \tilde{O}_\varepsilon(d)$, in contrast to Alman’s result that kicks in when k reaches about 2^d .

► **Theorem 3.** *Given $d \geq 2$ and $\varepsilon > 0$, there exists $\gamma = \Omega\left(\frac{1}{d^{3/2} \log^3(d)} \cdot \frac{\varepsilon^2}{\log^2(1/\varepsilon)}\right)$ such that the following holds for any sequence of matrices M_1, M_2, \dots, M_k where $M_i \in \mathbb{F}^{d_i \times d_i}$ for some $d_i \leq d$. Let $M = \bigotimes_{i=1}^k M_i$ and $n = \prod_{i=1}^k d_i$. If $n \geq d^{1/\gamma}$, then $R_{\mathbb{F}}(M, n^{1-\gamma}) \leq n^{1+\varepsilon}$.*

If the d_i are within a constant factor of each other, we obtain the following stronger result.

► **Theorem 4.** *Given $d \geq 2$, $\varepsilon > 0$, and a constant $c > 0$, there exists $\gamma = \Omega_c\left(\frac{1}{d \log d} \cdot \frac{\varepsilon^2}{\log^2(1/\varepsilon)}\right)$ such that the following holds for any sequence of matrices M_1, M_2, \dots, M_k where $M_i \in \mathbb{F}^{d_i \times d_i}$ and $cd \leq d_i \leq d$. Let $M = \bigotimes_{i=1}^k M_i$ and $n = \prod_{i=1}^k d_i$. If $n \geq d^{1/\gamma}$, then $R_{\mathbb{F}}(M, n^{1-\gamma}) \leq n^{1+\varepsilon}$.*

In the above theorem, the dependence of γ on c is nearly linear and the explicit formula can be found in Corollary 22.

The key strength of our results is that we do not need to assume uniform size of the matrices participating in the Kronecker product. Previous approaches depended on uniform size because of their reliance either on a polynomial method, or on an induction on the size of the matrix. As an applications of the bound for matrices of non-uniform size we show that our results significantly expand the class of Hadamard matrices that are known not to be Valiant-rigid. We show that the Kronecker products of Paley-Hadamard matrices and Hadamard matrices of bounded size are not Valiant-rigid (see Sec. 1.4).

Another strength of our improvement is that our bounds are sufficiently strong to be fed into the machinery developed by Dvir and Liu [6] for matrices of “well-factorable” size. Hence, we expect that this improvement might lead to further applications.

We note, that our bound on γ in Theorem 4 matches the Dvir–Liu bounds for Kronecker powers of specific classes of matrices, such as the generalized Walsh–Hadamard matrices and DFT matrices² of direct products of small abelian groups.

We also note that our upper bounds, similarly to upper bounds in recent work [2, 5, 6, 1], apply to a stronger notion of rigidity, called *row-column rigidity*.

► **Definition 5 (Row-column rigidity).** *For a matrix $A \in \mathbb{F}^{n \times n}$ and a target rank $0 \leq r \leq n$, let $R_{\mathbb{F}}^{rc}(A, r)$ be the minimal t for which there exists $Z \in \mathbb{F}^{n \times n}$ such that $\text{rank}(A - Z) \leq r$ and every row and column of Z has at most t non-zero entries.*

In Theorems 2, 3, and 4 the conclusion can be replaced by $R_{\mathbb{F}}^{rc}(M, n^{1-\gamma}) \leq n^\varepsilon$. Clearly, the latter statement is stronger, as for any $A \in \mathbb{F}^{n \times n}$ the inequality $R_{\mathbb{F}}(A, r) \leq n \cdot R_{\mathbb{F}}^{rc}(A, r)$ holds. The row-column versions of Theorems 3, and 4 are stated and proved as Theorem 24 and Corollary 23.

² The DFT (Discrete Fourier Transform) matrix of a finite abelian group G is the character table of G .

1.3 The field

We should point out that Theorems 2-4 make no assumption about the field \mathbb{F} , and they use elements of \mathbb{F} for the rank reduction. This is in contrast to the results of Dvir and Liu who require a field extension to achieve their rank reduction.

If the field \mathbb{F} is the field of definition of the matrix A , in [3] we call the corresponding rigidity function $R_{\mathbb{F}}(A, \cdot)$ the *strict rigidity* of A . If $\overline{\mathbb{F}}$ denotes the algebraic closure of \mathbb{F} then we call $R_{\overline{\mathbb{F}}}(A, \cdot)$ the *absolute rigidity* of A because, as shown in [3], this gives the smallest possible rigidity among all extension fields. A gap between these two quantities is demonstrated in [3]. Note that an upper bound on strict rigidity is a stronger statement than the same upper bound on absolute rigidity.

In this terminology, Dvir and Liu give upper bounds on absolute rigidity, whereas Alman's result and our results give upper bounds on strict rigidity.

1.4 Application of our results: rigidity upper bounds for Hadamard matrices

We recall that an *Hadamard matrix* is a square matrix whose entries are $+1$ and -1 and whose rows are mutually orthogonal. In addition to many other interesting properties, it was long believed that one can find a family of Valiant-rigid matrices among Hadamard matrices.

Contrary to expectations, two of the most well-studied families of Hadamard matrices were recently shown to be not Valiant-rigid. In 2017, Alman and Williams [2] proved that Walsh-Hadamard matrices are not strictly Valiant-rigid. As a corollary to [6], in [3] it was shown that Paley-Hadamard matrices are not absolutely Valiant-rigid. These results inspired the following conjecture.

► **Conjecture 6** (Babai). *The family of known Hadamard matrices is not strictly Valiant-rigid.*

We mention that in addition to infinite families, new classes of Hadamard matrices arise as Kronecker products of a steadily growing starter set of small Hadamard matrices with other known Hadamard matrices (see, e.g., surveys [8, 9]). Indeed, note that if H_1 and H_2 are Hadamard matrices, then $H_1 \otimes H_2$ is an Hadamard matrix as well.

As an application of our results for Kronecker products of matrices of non-uniform size we further expand the family of Hadamard matrices that are known not to be Valiant-rigid.

► **Theorem 7.** *Let \mathcal{F}_0 be the family of Paley-Hadamard matrices and Hadamard matrices of bounded size. Let \mathcal{F} be the family of all matrices that can be obtained as Kronecker products of some matrices from \mathcal{F}_0 . Then \mathcal{F} is not absolutely Valiant-rigid.*

► **Remark 8.** We note that Hadamard matrices are naturally defined over \mathbb{Q} , while the theorem above shows that matrices in \mathcal{F} are not sufficiently rigid when we make changes from \mathbb{C} . It is still open whether Paley-Hadamard matrices are strictly Valiant-rigid. If one proves that Paley-Hadamard matrices satisfy the inequality from Eq. (2), our proof will immediately yield the stronger version of the theorem above, that \mathcal{F} is not strictly Valiant-rigid.

► **Remark 9.** We note that Theorem 7 does not follow from Alman's original upper bound for rigidity of Kronecker products (Theorem 2).

A more general version of Theorem 7 can be stated for Kronecker products of matrices of bounded size and matrices that are sufficiently not rigid.

► **Theorem 10.** Let $0 < \varepsilon < 1/2$ and $b \geq 2$. Let \mathcal{F} be a family of matrices over \mathbb{F} , such that for every $d \times d$ matrix $A \in \mathcal{F}$ either $d \leq b$, or

$$R_{\mathbb{F}}^{rc}(A, d^{1-\gamma}) \leq d^\varepsilon \quad \text{for } \gamma = \frac{12(\log \log d)^2}{\varepsilon^3 \log d}. \quad (2)$$

Then, for every sequence of matrices $M_1, M_2, \dots, M_k \in \mathcal{F}$, the $n \times n$ matrix $M = \bigotimes_{i \in [k]} M_i$ either satisfies $R_{\mathbb{F}}^{rc}(M, n/\log n) \leq n^{6\varepsilon}$, or n is bounded above by a function of b and ε .

► **Corollary 11.** Let \mathcal{F} be a family of square matrices over \mathbb{F} . If for every $0 < \varepsilon < 1/2$ there exists $b \geq 2$ such that \mathcal{F} satisfies the assumptions of Theorem 10, then the family of Kronecker products of matrices from \mathcal{F} is not Valiant-rigid over \mathbb{F} .

We prove Theorems 7 and 10 in Section 5.

1.5 Our approach

In order to prove Theorem 2, Alman [1] first uses a beautiful trick to deduce the claim for Kronecker products of 2×2 matrices. He observes that it is sufficient to prove the claim for $R = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$. The Kronecker powers of R have low rigidity since they are very sparse. After that, he applies induction on the size d of the matrices involved in the Kronecker product. The technically involved induction argument leads to the factor 2^{-d} in γ .

Our proof is simpler and omits induction. Instead, we observe that an idea, somewhat similar to Alman’s proof of the base case $d = 2$, can be applied for any d .

Our key observation is that any $d \times d$ matrix can be written as a product of at most $2d$ very sparse matrices and $2d$ permutation matrices.

Specifically, for a vector $x \in \mathbb{F}^d$ define a $d \times d$ matrix

$$G_d(x) = \begin{pmatrix} I_{d-1} & x \\ 0 & \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \dots & 0 & x_1 \\ 0 & 1 & 0 \dots & 0 & x_2 \\ \dots & & & & \\ 0 & 0 & 0 \dots & 1 & x_{d-1} \\ 0 & 0 & 0 \dots & 0 & x_d \end{pmatrix}. \quad (3)$$

We are going to call the matrices of this form the *V-matrices* for the pattern of their non-zero entries. Next, it is not hard to verify that any $d \times d$ matrix can be written as

$$A = P_1 \cdot G_d(y)^T \cdot \begin{pmatrix} B & 0 \\ 0 & \lambda \end{pmatrix} \cdot G_d(x) \cdot P_2,$$

where $\lambda \in \{0, 1\} \subseteq \mathbb{F}$, $B \in \mathbb{F}^{(d-1) \times (d-1)}$ and P_1, P_2 are permutation matrices. By repeating this procedure for B at most $d - 2$ times one ends up with a product of $2d - 2$ V-matrices, a diagonal matrix, and permutation matrices (see Section 2.1).

As was observed in [6], the row-column rigidity of the product can be controlled by the row-column rigidity of each component.

► **Lemma 12.** For arbitrary $d \times d$ matrices A and B over a field \mathbb{F}

$$R_{\mathbb{F}}^{rc}(A \cdot B, r + s) \leq R_{\mathbb{F}}^{rc}(A, r) \cdot R_{\mathbb{F}}^{rc}(B, s). \quad (4)$$

Recall that a *monomial matrix* is a matrix where each row and each column has at most one non-zero element. A matrix is monomial exactly if it is the product of a diagonal matrix and a permutation matrix. A Kronecker product of monomial matrices is itself a monomial matrix. Moreover, if P is a monomial matrix then $R_{\mathbb{F}}^{rc}(P, 0) = 1$. Thus, in order to prove Theorem 4 one only needs to show strong upper bounds on row-column rigidity for Kronecker products of V-matrices.

To bound the row-column rigidity of a matrix M that is a Kronecker product of V-matrices one needs to observe that most of the non-zero entries are concentrated in just a few columns and rows, and so these entries form a low-rank matrix.

We discuss our strategy for upper bounds on rigidity of Kronecker products in Section 2. We prove Theorem 4 for the case of matrices of equal size in Section 3. We discuss rigidity bounds for matrices of unequal sizes in Section 4. Some standard tail bounds for binomial distributions are reviewed in Appendix A. We expand the family of Hadamard matrices known not to be Valiant rigid in Section 5. The proof of Obs. 14 omitted in Section 2 is provided in Appendix B.

1.6 Notation

We use $[n]$ to denote the set $\{1, 2, \dots, n\}$. \mathbb{F} denotes a field. Throughout the paper we use \vec{d} to denote the vector (d_1, d_2, \dots, d_k) , where each $d_i \geq 2$. We define $[\vec{d}] = [d_1] \times [d_2] \times \dots \times [d_k]$. We say that $X \in \mathbb{F}^{\vec{d}}$ if $X = (X_1, X_2, \dots, X_k)$, where $X_i \in \mathbb{F}^{d_i}$.

We use standard asymptotic notation. Let $f(n), g(n)$ be non-negative functions. We say that $f(n) = O(g(n))$ if there exists a constant C such that $f(n) \leq Cg(n)$ for all sufficiently large n . We say that $f(n) = \Omega(g(n))$ if $g(n) = O(f(n))$. Finally, $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

2 A strategy for upper bounds on the rigidity of Kronecker products

2.1 Expressing a matrix as a product of V-matrices

Recall that for a vector $x \in \mathbb{F}^d$ we define a $d \times d$ V-matrix

$$G_d(x) = \begin{pmatrix} I_{d-1} & x \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \dots & 0 & x_1 \\ 0 & 1 & 0 \dots & 0 & x_2 \\ \dots & & & & \\ 0 & 0 & 0 \dots & 1 & x_{d-1} \\ 0 & 0 & 0 \dots & 0 & x_d \end{pmatrix}. \quad (5)$$

Let $\vec{d} = (d_1, d_2, \dots, d_k)$. For $X = (X_1, X_2, \dots, X_k)$ with $X_i \in \mathbb{F}^{d_i}$ for $i \in [k]$, define

$$G_{\vec{d}}(X) = \bigotimes_{i=1}^k G_{d_i}(X_i). \quad (6)$$

In the case when \vec{d} consists of k equal coordinates $d_i = d$, we use notation $G_{d,k}(X) := G_{\vec{d}}(X)$.

► **Definition 13.** *A square matrix P is called a permutation matrix, if every row and every column of P has precisely one non-zero entry, which is equal to 1.*

Recall, that arbitrary permutation of columns (rows) of a matrix can be represented by right (left, respectively) multiplication by a permutation matrix.

The multiplication of an $m \times d$ matrix A by $G_d(x)$ from the right corresponds to keeping the first $d - 1$ columns of A unchanged and replacing the last column of A with a linear combination of the columns of A with the coefficients given by x . We make the following observation.

► **Observation 14.** For any matrix $A \in \mathbb{F}^{d \times d}$ there exist $B \in \mathbb{F}^{(d-1) \times (d-1)}$, vectors $x, y \in \mathbb{F}^d$, $\lambda \in \{0, 1\} \subseteq \mathbb{F}$ and permutation matrices $P_1, P_2 \in \mathbb{F}^{d \times d}$ such that

$$A = P_1 \cdot G_d(y)^T \cdot \begin{pmatrix} B & 0 \\ 0 & \lambda \end{pmatrix} \cdot G_d(x) \cdot P_2. \quad (7)$$

Proof. See Appendix B. ◀

► **Corollary 15.** For any matrix $A \in \mathbb{F}^{d \times d}$ there exist a diagonal matrix W , $2(d - 1)$ vectors X_i, Y_i for $i \in [d - 1]$, and $2(d - 1)$ permutation matrices P_i, Q_i for $i \in [d - 1]$ such that

$$A = Q_1 \cdot G_d(Y_1)^T \cdot Q_2 \cdot \dots \cdot G_d(Y_{d-1})^T \cdot W \cdot G_d(X_{d-1}) \cdot P_{d-1} \cdot \dots \cdot G_d(X_1) \cdot P_1. \quad (8)$$

Proof. Let $t \in [d - 1]$. Observe that for $x \in \mathbb{F}^t$ and the vector $y \in \mathbb{F}^d$, whose first $d - t$ coordinates are 0, and last t coordinates are equal to the corresponding coordinates of x ,

$$\begin{pmatrix} G_{d-t}(x) & 0 \\ 0 & I_t \end{pmatrix} = P_1 G_d(y) P_2, \quad (9)$$

where I_t is the $t \times t$ identity matrix and P_1, P_2 are the permutation matrices that exchange the first $d - t$ and the last t rows and columns, respectively.

Then, the claim of the corollary follows from Observation 14 by applying it recursively $d - 1$ times to the remaining top-left block B , until we are left with a diagonal matrix. ◀

By combining the above corollary with Lemma 12 we obtain the following inequality.

► **Lemma 16.** Let $d \geq 2$, $r \leq d^k$ and $M_1, M_2, \dots, M_k \in \mathbb{F}^{d \times d}$. Then

$$R_{\mathbb{F}}^{rc} \left(\bigotimes_{i=1}^k M_i, (2d - 2)r \right) \leq \left(\max_{X \in \mathbb{F}^{d \times k}} R_{\mathbb{F}}^{rc} (G_{d,k}(X), r) \right)^{2d-2}. \quad (10)$$

Proof. By Corollary 15, for each $i \in [k]$ there exist $2(d - 1)$ vectors $X_i^{(j)}, Y_i^{(j)} \in \mathbb{F}^d$, $2(d - 1)$ permutation matrices $P_i^{(j)}, Q_i^{(j)}$ and a diagonal matrix W_i such that

$$M_i = \prod_{j=1}^{d-1} \left(Q_i^{(j)} \cdot G_d(Y_i^{(j)})^T \right) \cdot W_i \cdot \prod_{j=1}^{d-1} \left(G_d(X_i^{(d-j)}) \cdot P_i^{(d-j)} \right).$$

Then, using that $(A_1 \otimes B_1) \cdot (A_2 \otimes B_2) = (A_1 A_2) \otimes (B_1 B_2)$ holds for any matrices, we get

$$\bigotimes_{i=1}^k M_i = \prod_{j=1}^{d-1} \left(\bigotimes_{i=1}^k Q_i^{(j)} \bigotimes_{i=1}^k G_d(Y_i^{(j)})^T \right) \cdot \left(\bigotimes_{i=1}^k W_i \right) \cdot \prod_{j=1}^{d-1} \left(\bigotimes_{i=1}^k G_d(X_i^{(d-j)}) \bigotimes_{i=1}^k P_i^{(d-j)} \right).$$

Let $X^{(j)} \in \mathbb{F}^{d \times k}$ be a matrix whose i -th column is $X_i^{(j)}$. Define $Y^{(j)} \in \mathbb{F}^{d \times k}$ similarly. Let

$$Q^{(j)} = \bigotimes_{i=1}^k Q_i^{(j)}, \quad P^{(j)} = \bigotimes_{i=1}^k P_i^{(j)}, \quad W = \bigotimes_{i=1}^k W_i. \quad (11)$$

68:8 Improved Upper Bounds for the Rigidity of Kronecker Products

Then, $Q^{(j)}$ and $P^{(j)}$ are permutation matrices, and W is a diagonal matrix, and

$$M = \bigotimes_{i=1}^k M_i = \prod_{j=1}^{d-1} \left(Q^{(j)} \cdot G_{d,k}(Y^{(j)})^T \right) \cdot W \cdot \prod_{j=1}^{d-1} \left(G_{d,k}(X^{(d-j)}) \cdot P^{(d-j)} \right). \quad (12)$$

Thus, by Lemma 12,

$$\begin{aligned} R_{\mathbb{F}}^{rc}(M, (2d-2)r) &\leq \prod_{j=1}^{d-1} R_{\mathbb{F}}^{rc}(G_{d,k}(Y^{(j)}), r) \cdot \prod_{j=1}^{d-1} R_{\mathbb{F}}^{rc}(G_{d,k}(X^{(j)}), r) \leq \\ &\leq \left(\max_{X \in \mathbb{F}^{d \times k}} R_{\mathbb{F}}^{rc}(G_{d,k}(X), r) \right)^{2d-2}. \end{aligned} \quad (13)$$

◀

Furthermore, we note that a similar statement holds for Kronecker products of matrices of not necessarily equal size.

► **Lemma 17.** *Let $d_i \geq 2$ and $M_i \in \mathbb{F}^{d_i \times d_i}$ for $i \in [k]$. Assume $d_k \geq d_i$ for $i \in [k]$. Then*

$$R_{\mathbb{F}}^{rc} \left(\bigotimes_{i=1}^k M_i, (2d_k - 2)r \right) \leq \left(\max_{X \in \mathbb{F}^{\vec{d}}} R_{\mathbb{F}}^{rc}(G_{\vec{d}}(X), r) \right)^{2d_k - 2}. \quad (14)$$

Proof. For M_i with $d_i < d_k$ we multiply the decomposition given by Corollary 15, by $2(d_k - d_i)$ identity matrices from the left and from the right, so that every M_i is decomposed into the product of precisely $4d_k - 3$ matrices. After this, the proof is identical to the proof of the previous lemma. ◀

2.2 General approach to upper bounds for the rigidity of $G_{\vec{d}}(X)$

As we see from Lemmas 16 and 17, in order to prove Theorems 3 and 4 it is sufficient to bound the row-column rigidity of Kronecker products of V-matrices. We describe a general approach to such bounds.

The key observation is that most of the non-zero entries of $G_{\vec{d}}(X)$ are concentrated in just a few columns and a few rows. Hence, after deleting these columns and rows we expect to get a matrix with very sparse rows and columns.

Recall, that for $\vec{d} = (d_1, d_2, \dots, d_k)$ we define $[\vec{d}] = [d_1] \times [d_2] \times \dots \times [d_k]$. The rows and the columns of $G_{\vec{d}}(X)$ are indexed by tuples in $[\vec{d}]$.

For a $d_i \times d_i$ V-matrix, all rows, except the d_i -th row, have up to two non-zero entries, while the d_i -th row has only one non-zero entry. For V-matrices, all columns, except the last one, have one non-zero entry and the last column may have up to d_i non-zero entries. Therefore, the densest rows are indexed by tuples with a small number of i -th coordinates being equal d_i and the densest columns are indexed by tuples with a large number of i -th coordinates being equal d_i , for $i \in [k]$.

Let $w : \mathbb{R} \rightarrow (0, \infty)$ be a function. Define a score of a string $x \in [\vec{d}]$ as

$$s(x) = \sum_{i=1}^k w(d_i) \mathbf{1}[x_i = d_i]. \quad (15)$$

Consider the uniform distribution on $[\vec{d}]$. Then $\chi(i) = w(d_i)\mathbf{1}[x_i = d_i]$ are independent random variables, and $s(x) = \sum_{i=1}^k \chi(i)$ with

$$\mathbb{E}_x[s(x)] = \sum_{i=1}^k \frac{w(d_i)}{d_i} \quad \text{and} \quad \text{Var}[s(x)] = \sum_{i=1}^k w(d_i)^2 \left(\frac{d_i - 1}{d_i^2} \right). \quad (16)$$

For $\delta > 0$ we define a pair of sets of strings of high and low scores, respectively

$$\mathcal{C}(\delta) = \{x \in [\vec{d}] \mid s(x) \geq \mathbb{E}_y[s(y)] + \delta\} \quad \text{and} \quad \mathcal{R}(\delta) = \{x \in [\vec{d}] \mid s(x) \leq \mathbb{E}_y[s(y)] - \delta\}. \quad (17)$$

These sets correspond to indices of the most dense columns and rows, respectively. Note, that one may use concentration inequalities to bound the sizes of these sets (see Appendix A). The matrix E defined by the union of the rows of $G_{\vec{d}}(X)$ in $\mathcal{R}(\delta)$ and the columns of $G_{\vec{d}}(X)$ in $\mathcal{C}(\delta)$ has low rank. Now we want to count the number of non-zero entries in rows and columns of $G_{\vec{d}}(X) - E$. Define

$$T_c(\delta, y) = \{x \in [\vec{d}] \setminus \mathcal{R}(\delta) \mid \forall i : y_i \in \{x_i, d_i\}\} \quad \text{and} \quad M_c(\delta) = \max_{y \in [\vec{d}] \setminus \mathcal{C}(\delta)} |T_c(\delta, y)|; \quad (18)$$

$$T_r(\delta, x) = \{y \in [\vec{d}] \setminus \mathcal{C}(\delta) \mid \forall i : y_i \in \{x_i, d_i\}\} \quad \text{and} \quad M_r(\delta) = \max_{x \in [\vec{d}] \setminus \mathcal{R}(\delta)} |T_r(\delta, x)|. \quad (19)$$

With this notation, we have the following inequality.

► **Lemma 18.** *Let $\delta > 0$. Then, for any $X \in \mathbb{F}^{\vec{d}}$,*

$$R_{\mathbb{F}}^{rc}(G_{\vec{d}}(X), |\mathcal{C}(\delta)| + |\mathcal{R}(\delta)|) \leq \max(M_c(\delta), M_r(\delta)).$$

Proof. Let E be the matrix obtained from $G_{\vec{d}}(X)$ by changing to 0 every entry that is not in a column with index in $\mathcal{C}(\delta)$ and is not in a row with index in $\mathcal{R}(\delta)$. Then

$$\text{rank}(E) \leq |\mathcal{C}(\delta)| + |\mathcal{R}(\delta)|. \quad (20)$$

Let $Z = G_{\vec{d}}(X) - E$ and $x, y \in [\vec{d}]$. Observe that the entry of $G_{\vec{d}}(X)$ with coordinates (x, y) is non-zero only if for every $i \in [k]$ either $x_i = y_i$ or $y_i = d_i$. Therefore, every row of Z has at most $M_r(\delta)$ non-zero entries, and every column of Z has at most $M_c(\delta)$ non-zero entries. ◀

Hence, in order to prove an upper bound on the rigidity of $G_{\vec{d}}(X)$, one just may come up with a good choice of weights w and a threshold $\delta > 0$ which make all the quantities $|\mathcal{C}(\delta)|$, $|\mathcal{R}(\delta)|$, $M_c(\delta)$ and $M_r(\delta)$ small.

3 Rigidity of Kronecker products of matrices of uniform size

Now, we show how the approach, described above, provides a strong upper bound on the rigidity of Kronecker products of matrices of uniform size. We follow the notation introduced earlier.

► **Theorem 19.** *Given $d \geq 2$ and $0 < \varepsilon < 1$, there exists $\gamma = \Theta\left(\frac{1}{d \log d} \cdot \frac{\varepsilon^2}{\log^2(1/\varepsilon)}\right)$ such that for all $k \geq 1$ and $X \in \mathbb{F}^{d \times k}$ we have $R_{\mathbb{F}}^{rc}(G_{d,k}(X), n^{1-\gamma}) \leq n^{\varepsilon/d}$, where $n = d^k$.*

68:10 Improved Upper Bounds for the Rigidity of Kronecker Products

Proof. We use the approach described in the previous section. We take $w(x) = 1$, then $s(x)$ given by Eq. (15) counts the number of coordinates equal to d . A simple computation gives

$$\mathbb{E}[s(x)] = k/d \quad \text{and} \quad \text{Var}[s(x)] = k(d-1)/d^2. \quad (21)$$

Then, by the Bernstein inequality (see Corollary 32), for $\delta < 1/d$,

$$|\mathcal{C}(\delta k)| = d^k \cdot \mathbb{P}[s(x) \geq k/d + \delta k] \leq \exp\left(\left(\ln d - \frac{\delta^2 d}{3}\right)k\right). \quad (22)$$

$$|\mathcal{R}(\delta k)| = d^k \cdot \mathbb{P}[s(x) \leq k/d - \delta k] \leq \exp\left(\left(\ln d - \frac{\delta^2 d}{3}\right)k\right). \quad (23)$$

Therefore, for $\delta = \Theta\left(\frac{\varepsilon}{\log(1/\varepsilon)d}\right)$,

$$|\mathcal{R}(\delta k)| + |\mathcal{C}(\delta k)| \leq d^{(1-\gamma)k} = n^{1-\gamma}, \quad \text{for some } \gamma = \Theta\left(\frac{\varepsilon^2}{d \log d \log^2(1/\varepsilon)}\right). \quad (24)$$

For a string $x \in [d]^k$ define the set $S_x = \{i \mid x_i = d\}$. Recall, that for every $y \in [d]^k \setminus \mathcal{C}(\delta k)$ we have $|S_y| \leq k(1/d + \delta)$ and for every $x \in [d]^k \setminus \mathcal{R}(\delta k)$ we have $|S_x| \geq k(1/d - \delta)$.

For each $y \in [d]^k \setminus \mathcal{C}(\delta k)$, a vector $x \in T_c(\delta k, y)$ can be described by picking a subset U of S_y of the size at most $2\delta k$ such that $U = S_y \setminus S_x$ and by picking $x_j \in [d-1]$ for every $j \in U$. So, by Lemma 33,

$$|T_c(\delta k, y)| \leq \sum_{i \leq 2\delta k} \binom{|S_y|}{i} (d-1)^i \leq \left(\frac{e|S_y|(d-1)}{2\delta k}\right)^{2\delta k} \leq \exp(2\delta \ln(3/\delta)k). \quad (25)$$

Similarly, for every row index $x \in [d]^k \setminus \mathcal{R}(\delta k)$, a column $y \in T_r(\delta k, x)$ can be described by picking a subset $U \subseteq [k] \setminus S_x$ of size at most $2\delta k$ and by setting $y_j = d$ for $j \in U$ and $y_j = x_j$ for $j \notin U$. Hence, by Lemma 33,

$$|T_r(\delta k, x)| \leq \sum_{i \leq 2\delta k} \binom{k - |S_x|}{i} \leq \left(\frac{ek}{2\delta k}\right)^{2\delta k} \leq \exp(2\delta \ln(2/\delta)k). \quad (26)$$

Therefore,

$$\max(M_c(\delta k), M_r(\delta k)) \leq \exp(2\delta \ln(3/\delta)k) \leq d^{\varepsilon k/d}, \quad \text{for some } \delta = \Theta\left(\frac{\varepsilon}{\log(1/\varepsilon)d}\right). \quad (27)$$

Hence, the conclusion of the theorem follows from Lemma 18. \blacktriangleleft

Finally, we can deduce our improved bound for the rigidity of Kronecker products of matrices of uniform size.

► Theorem 20. *Given $d \geq 2$ and $0 < \varepsilon < 1$, there exists $\gamma = \Omega\left(\frac{1}{d \log d} \cdot \frac{\varepsilon^2}{\log^2(1/\varepsilon)}\right)$ such that the following holds for any $M_1, M_2, \dots, M_k \in \mathbb{F}^{d \times d}$ with $k > 1/\gamma$. Let $M = \bigotimes_{i=1}^k M_i$ and $n = d^k$. Then*

$$R_{\mathbb{F}}^{rc}(M, n^{1-\gamma}) \leq n^{\varepsilon}, \quad \text{and so} \quad R_{\mathbb{F}}(M, n^{1-\gamma}) \leq n^{1+\varepsilon}.$$

Proof. By Theorem 19 and Lemma 16, for $\varepsilon' = \varepsilon/2$ we have

$$R_{\mathbb{F}}^{rc}(M, 2d \cdot d^{(1-\gamma')k}) \leq d^{2\varepsilon'k} \quad \text{for some } \gamma' = \Theta\left(\frac{\varepsilon^2}{\log^2(1/\varepsilon)d \log d}\right).$$

Pick $\gamma = \gamma'/3$ and note that for $k > 1/\gamma$ we have $2d \cdot d^{-2\gamma k} \leq 2d \cdot d^{-2} \leq 1$. \blacktriangleleft

4 Rigidity bounds for matrices of unequal sizes

In this section we show how a similar approach may be used to show upper bounds on the rigidity of Kronecker products of matrices of not necessarily equal sizes.

► **Theorem 21.** *Let $0 < \varepsilon < 1$ and let $w(x) \geq 1$ be a non-decreasing function. There exists an absolute constant $c > 0$ such that the following is true. Let $2 \leq d_1 \leq d_2 \leq \dots \leq d_k$ be integers. Let $L > 0$ and $K > 0$ be such that*

- (i) $\sum_{i=1}^k w(d_i)/d_i \leq K \left(\sum_{i=1}^k \log d_i / \log d_k \right) \cdot (w(d_1)/d_k)$, and
- (ii) $\sum_{i=1}^k w(d_i)/d_i \geq L \left(\sum_{i=1}^k \log d_i / \log d_k \right) \cdot (w(d_k)/d_k)$.

Let $n = \prod_{i=1}^k d_i$. Consider arbitrary matrices $M_i \in \mathbb{F}^{d_i \times d_i}$ and let $M = \bigotimes_{i=1}^k M_i$. Then

$$R_{\mathbb{F}}^{rc}(M, 2d_k \cdot n^{1-\gamma}) \leq n^\varepsilon \quad \text{where} \quad \gamma = \frac{cL\varepsilon^2}{d_k \log d_k \cdot K^2 \log^2(K/\varepsilon)}.$$

Proof. We follow the notation of Section 2.2. Recall that

$$m := \mathbb{E}[s(x)] = \sum_{i=1}^k w(d_i)/d_i \quad \text{and} \quad \text{Var}[s(x)] = \sum_{i=1}^k w(d_i)^2 \left(\frac{d_i - 1}{d_i^2} \right) \leq w(d_k)m. \quad (28)$$

Then, by the Bernstein inequality, for $0 < \delta < 1$,

$$\begin{aligned} |\mathcal{C}(\delta m)| &= n \cdot \mathbb{P}[s(x) \geq m + \delta m] \leq n \cdot \exp\left(-\frac{\delta^2 m^2 / 2}{\text{Var}[s(x)] + \delta w(d_k)m/3}\right) \leq \\ &\leq n \cdot \exp\left(-\frac{\delta^2 m}{3w(d_k)}\right). \end{aligned} \quad (29)$$

and similarly,

$$|\mathcal{R}(\delta m)| = n \cdot \mathbb{P}[s(x) \leq m - \delta m] \leq n \cdot \exp\left(-\frac{\delta^2 m}{3w(d_k)}\right). \quad (30)$$

Now we want to bound the number of entries in $T_c(\delta m, y)$ and $T_r(\delta m, x)$. For $x \in [d]$ define $S_x = \{i \in [n] \mid x_i = d_i\}$.

Let $x \in [d] \setminus \mathcal{R}(\delta m)$ and $y \in T_r(\delta m, x)$, by definition, $S_x \subseteq S_y$ and

$$\sum_{i \in S_y \setminus S_x} w(d_i) \leq 2\delta m. \quad (31)$$

Thus, $t := |S_y \setminus S_x| \leq 2\delta m/w(d_1) =: t_{\max}$. Assumption (i) implies that

$$t_{\max} \leq 2\delta K \ln(n)/(d_k \ln d_k).$$

At the same time, $t_{\max} \geq 2\delta \cdot (kw(d_1)/d_k)/w(d_1) = 2\delta k/d_k$. Note that $y \in T_r(\delta m, x)$ is uniquely defined by $S_y \setminus S_x$. Thus, by Lemma 33,

$$\begin{aligned} T_r(\delta m, x) &\leq \sum_{i=0}^{t_{\max}} \binom{k}{i} \leq \exp\left(t_{\max} \ln\left(\frac{ek}{t_{\max}}\right)\right) \leq \\ &\leq \exp(\ln n \cdot 2\delta K \ln(2d_k/\delta)/(d_k \ln d_k)). \end{aligned} \quad (32)$$

68:12 Improved Upper Bounds for the Rigidity of Kronecker Products

Similarly, for $y \in [\vec{d}] \setminus \mathcal{C}(\delta m)$ we have $x \in T_c(\delta m, y)$ only if $S_x \subseteq S_y$ and Eq. (31) is satisfied. Clearly, $|S_y| \leq m(1 + \delta)/w(d_1) \leq 2m/w(d_1)$, and $t = |S_y \setminus S_x|$ satisfies $t \leq t_{\max}$. Hence, by Lemma 33,

$$\begin{aligned} T_c(\delta m, y) &\leq \sum_{i=0}^{t_{\max}} \binom{|S_y|}{t_{\max}} d_k^i \leq d_k^{t_{\max}} \sum_{i=0}^{t_{\max}} \binom{2m/w(d_1)}{t_{\max}} \\ &\leq \exp(t_{\max} \ln(3\delta^{-1}) + t_{\max} \ln d_k) \leq \exp(\ln n \cdot 2\delta K \ln(3\delta^{-1})/d_k). \end{aligned} \quad (33)$$

Therefore, there exists $\delta = \Theta((\varepsilon/K)/\log(K/\varepsilon))$ such that

$$\max(M_r(\delta m), M_c(\delta m)) \leq n^{\varepsilon/(2d_k)}.$$

Moreover, for such choice of δ , by assumption (ii) and Eq. (29)-(30), we obtain

$$|\mathcal{C}(\delta m)| + |\mathcal{R}(\delta m)| \leq n \exp(-\delta^2 L \ln(n)/(3d_k \ln d_k)) \leq n^{1-\gamma}. \quad (34)$$

Thus, the claim follows from Lemma 18 and Lemma 17. \blacktriangleleft

Observe that Theorem 20 is a special case of Theorem 21 with $K = L = 1$. More generally, we can get the same bound on γ for the case when all the d_i are within a constant factor of each other.

► **Corollary 22.** *Given $0 < \varepsilon, c \leq 1$, and $d \geq 2$, there exists $\gamma = \Omega\left(\frac{c\varepsilon^2}{d \log(d) \log(1/c) \log^2(1/(c\varepsilon))}\right)$ such that the following holds for any sequence of matrices M_1, M_2, \dots, M_k where $M_i \in \mathbb{F}^{d_i \times d_i}$ and $cd \leq d_i \leq d$. Let $M = \bigotimes_{i=1}^k M_i$ and $n = \prod_{i=1}^k d_i$. If $n \geq d^{1/\gamma}$, then $R_{\mathbb{F}}^{rc}(M, n^{1-\gamma}) \leq n^\varepsilon$.*

Proof. Observe that the assumptions of Theorem 21 are satisfied for $w(x) = 1$ with $K = L \leq (1/c) \log(1/c)$. Let γ' be the constant provided by Theorem 21. We take $\gamma = \gamma'/3$ and note that $n^{2\gamma} \geq 2d$. Hence, $R_{\mathbb{F}}^{rc}(M, n^{1-\gamma}) \leq n^\varepsilon$. \blacktriangleleft

Next, we eliminate the lower bound constraint on the d_i , at the cost of a slightly worse dependence of γ on d . (Theorem 24 below). In preparation for proving Theorem 24, we state its special case where all but at most one of the d_i are restricted to the interval $[\sqrt{d}, d]$.

► **Corollary 23.** *Given $d \geq 2$ and $0 < \varepsilon < 1$, there exists $\gamma = \Omega\left(\frac{\varepsilon^2}{d^{3/2} \log(d) \log^2(d/\varepsilon)}\right)$ such that the following holds. Consider any $d_i \leq d$ and $M_i \in \mathbb{F}^{d_i \times d_i}$ for $i \in [k]$. Let $M = \bigotimes_{i=1}^k M_i$ and $n = \prod_{i=1}^k d_i$. Assume that $d_i \geq \sqrt{d}$ for all $i \geq 2$. If $n \geq d^{1/\gamma}$. Then $R_{\mathbb{F}}^{rc}(M, n^{1-\gamma}) \leq n^\varepsilon$.*

Proof. Define $w(d_i) = 1$. Then, the assumptions of Theorem 21 are satisfied for $K = L \leq 4\sqrt{d}$. Indeed, for $d_k \leq d$ and $d_2 \geq \sqrt{d}$ and $k > d$, we have the following trivial bounds

$$\sum_{i=1}^k \log(d_i)/\log(d_k) \geq (k-1)/2 \quad \text{and} \quad \sum_{i=1}^k 1/d_i \leq (k-1)/\sqrt{d} + 1 \leq 2(k-1)/\sqrt{d}. \quad (35)$$

Hence, by Theorem 21, there exists $\gamma' = \Omega\left(\frac{\varepsilon^2}{d^{3/2} \log(d) \log^2(d/\varepsilon)}\right)$ such that

$$R_{\mathbb{F}}^{rc}(M, 2dn^{1-\gamma'}) \leq n^\varepsilon.$$

Take $\gamma = \gamma'/3$. Finally, note that the assumption $n \geq d^{1/\gamma}$ implies that $n^{2\gamma} \geq 2d$ and $k \geq d$. Hence, $R_{\mathbb{F}}^{rc}(M, n^{1-\gamma}) \leq n^\varepsilon$. ◀

Finally, we prove our main result by eliminating $d_i \geq \sqrt{d}$ constraint using a bin packing argument.

► **Theorem 24.** *Given $d \geq 2$ and $0 < \varepsilon < 1$, there exists $\gamma = \Omega\left(\frac{\varepsilon^2}{d^{3/2} \log(d) \log^2(d/\varepsilon)}\right)$ such that the following holds for any sequence of matrices M_1, M_2, \dots, M_k where $M_i \in \mathbb{F}^{d_i \times d_i}$ and $2 \leq d_i \leq d$. Let $M = \bigotimes_{i=1}^k M_i$ and $n = \prod_{i=1}^k d_i$. If $n \geq d^{1/\gamma}$, then $R_{\mathbb{F}}^{rc}(M, n^{1-\gamma}) \leq n^\varepsilon$.*

Proof. Let us split the d_i into the smallest possible number k' of bins, such that the product of numbers in each bin is at most d . Let a_j be the product of the numbers in bin $j \in [k']$ and let A_j be the Kronecker product of the corresponding matrices. Then at most one a_i is $\leq \sqrt{d}$. Hence, the theorem follows from Corollary 23. ◀

5 Application to rigidity upper bounds for Hadamard matrices

As an application of our improved bounds, we show that the family of Kronecker products of sufficiently not rigid matrices and matrices of bounded size is not Valiant-rigid.

► **Theorem 25.** *Let $0 < \varepsilon < 1/2$ and $b \geq 2$. Let \mathcal{F} be a family of matrices over \mathbb{F} , such that for every $d \times d$ matrix $A \in \mathcal{F}$ either $d \leq b$, or*

$$R_{\mathbb{F}}^{rc}(A, d^{1-\gamma}) \leq d^\varepsilon \quad \text{for } \gamma = \frac{12(\log \log d)^2}{\varepsilon^3 \log d}. \quad (36)$$

Then, for every sequence of matrices $M_1, M_2, \dots, M_k \in \mathcal{F}$, the $n \times n$ matrix $M = \bigotimes_{i \in [k]} M_i$ either satisfies $R_{\mathbb{F}}^{rc}(M, n/\log n) \leq n^{6\varepsilon}$, or n is bounded above by a function of b and ε .

As an immediate application of Theorem 25, we significantly expand the family of Hadamard matrices known to be not Valiant rigid (Theorem 7). We rely on the rigidity upper bound for Paley-Hadamard matrices established in [6, 3].

► **Theorem 26** ([6, 3]). *There exist constants $c_1 > 0$ and $c_2 > 0$ such that for all $\varepsilon > 0$ and an arbitrary $d \times d$ Paley-Hadamard matrix A we have $R_{\mathbb{C}}^{rc}\left(A, \frac{d}{\exp(\varepsilon^{c_1} (\log d)^{c_2})}\right) \leq d^\varepsilon$.*

We restate Theorem 7 for convenience.

► **Theorem 27.** *Let \mathcal{F}_0 be the family of Paley-Hadamard matrices and Hadamard matrices of bounded size. Let \mathcal{F} be the family of all matrices that can be obtained as Kronecker products of some matrices from \mathcal{F}_0 . Then \mathcal{F} is not absolutely Valiant-rigid.*

Proof. Fix $\varepsilon > 0$. By Theorem 26, the inequality

$$\gamma = \frac{\varepsilon^{c_1}}{(\log d)^{1-c_2}} \geq \frac{12(\log \log d)^2}{\varepsilon^3 \log d} \Leftrightarrow \frac{(\log d)^{c_2}}{(\log \log d)^2} \geq 12\varepsilon^{-3-c_1}$$

holds for all sufficiently large d . Thus, the claim of the theorem follows from Theorem 25. ◀

68:14 Improved Upper Bounds for the Rigidity of Kronecker Products

In order to prove Theorem 25, we use the following inequality that relates the row-column rigidity of the Kronecker product of a pair of matrices to the row-column rigidities of each of the matrices participating in the product.

► **Lemma 28** (Dvir, Liu [6, Lemma 4.9]). *Let A be an $n \times n$ matrix and B be an $m \times m$ matrix. Then*

$$R_{\mathbb{F}}^{rc}(A \otimes B, r_a m + r_b n) \leq R_{\mathbb{F}}^{rc}(A, r_a) \cdot R_{\mathbb{F}}^{rc}(B, r_b)$$

In the pair of lemmas below we show that the Kronecker product of arbitrary many sufficiently large and sufficiently not rigid matrices is sufficiently not rigid itself. These lemmas are essentially the proof content of Lemma 4.10 in [6], which was stated and proved for more specific needs.

► **Lemma 29.** *Let $2 \leq b \leq d_1 \leq \dots \leq d_k \leq b^2$ be integers. Let $0 < \varepsilon < 1$ and $\gamma > 0$ such that $\gamma \log(b) \geq 2 \log(3/\varepsilon)$ and $b \geq 3/\varepsilon$. Assume that for every $i \in [k]$, M_i is a $d_i \times d_i$ matrix over \mathbb{F} that satisfies*

$$R_{\mathbb{F}}^{rc}(M_i, d_i^{1-\gamma}) \leq d_i^\varepsilon.$$

Define $M = \bigotimes_{i \in [k]} M_i$ and $n = \prod_{i \in [k]} d_i$. Then,

$$R_{\mathbb{F}}^{rc}(M, n^{1-\varepsilon\gamma/4}) \leq n^{4\varepsilon}.$$

Proof. By the assumptions of the lemma, we can write $M_i = A_i + E_i$, where $A_i, E_i \in \mathbb{F}^{d_i \times d_i}$, $\text{rank}(A_i) \leq d_i^{1-\gamma}$, and E_i has at most d_i^ε non-zero entries in every row and column. Then

$$\bigotimes_{i \in [k]} M_i = \bigotimes_{i \in [k]} (A_i + E_i) = \sum_{S \subseteq [k]} \bigotimes_{i \in S} A_i \bigotimes_{j \in [k] \setminus S} E_j \quad (37)$$

Now all the summands can be split into two groups: when $|S| \geq \varepsilon k$ and when $|S| < \varepsilon k$. We are going to bound the rank of the sum of the first group and the number of non-zero entries of the sum of the second group.

$$\begin{aligned} \text{rank} \left(\sum_{S \subseteq [k], |S| \geq \varepsilon k} \bigotimes_{i \in S} A_i \bigotimes_{j \in [k] \setminus S} E_j \right) &\leq \sum_{S \subseteq [k], |S| = \varepsilon k} \text{rank} \left(\bigotimes_{i \in S} A_i \right) \prod_{j \in [k] \setminus S} d_j \leq \\ &\leq \binom{k}{\varepsilon k} \max_{S \subseteq [k], |S| = \varepsilon k} \prod_{i \in S} d_i^{1-\gamma} \prod_{j \in [k] \setminus S} d_j \leq \left(\frac{3}{\varepsilon} \right)^{\varepsilon k} \cdot n \cdot b^{-\varepsilon k \gamma} \end{aligned} \quad (38)$$

Since $b^\gamma \geq (3/\varepsilon)^2$ and $n \leq b^{2k}$, we have

$$\text{rank} \left(\sum_{S \subseteq [k], |S| \geq \varepsilon k} \bigotimes_{i \in S} A_i \bigotimes_{j \in [k] \setminus S} E_j \right) \leq n \cdot b^{-\varepsilon k \gamma / 2} \leq n^{1-\varepsilon\gamma/4} \quad (39)$$

Next, consider the remaining terms

$$E = \sum_{S \subseteq [k], |S| < \varepsilon k} \bigotimes_{i \in S} A_i \bigotimes_{j \in [k] \setminus S} E_j. \quad (40)$$

Using Lemma 33, every column and every row of E has at most

$$\left(\sum_{i=0}^{\varepsilon k} \binom{k}{i} \right) (b^2)^{\varepsilon k} \prod_{i \in [k]} d_i^\varepsilon \leq \left(\frac{3}{\varepsilon} \right)^{\varepsilon k} n^{2\varepsilon} n^\varepsilon \leq b^{\varepsilon k} n^{3\varepsilon} \leq n^{4\varepsilon} \quad (41)$$

non-zero entries. ◀

► **Lemma 30.** *Let $2 \leq b \leq d_1 \leq \dots \leq d_k$ be integers. Let $0 < \varepsilon < 1$ and $\gamma_1, \gamma_2, \dots, \gamma_k > 0$ be such that $b \geq 3/\varepsilon$ and for all $i \in [k]$ we have $\gamma_i \cdot \log d_i \geq 4 \log(3/\varepsilon)$. Assume that for every $i \in [k]$, M_i is a $d_i \times d_i$ matrix over \mathbb{F} , that satisfies $R_{\mathbb{F}}^{rc}(M_i, d_i^{1-\gamma_i}) \leq d_i^\varepsilon$.*

Define $M = \bigotimes_{i \in [k]} M_i$ and $n = \prod_{i \in [k]} d_i$. Then,

$$R_{\mathbb{F}}^{rc}(M, n^{1-\psi}) \leq n^{5\varepsilon} \quad \text{for} \quad \psi = \frac{\varepsilon^2 \min_i \gamma_i}{4 \log \log d_k} - \frac{\log \log \log d_k}{\log n}$$

Proof. Let $I_t = \{i \mid d_i \in (b^{2^t}, b^{2^{t+1}}]\}$ for $t = 1, \dots, L = \log \log d_k$. Let $A_t = \bigotimes_{i \in I_t} M_i$ and $n_t = \prod_{i \in I_t} d_i$. Let $\gamma = \min_i \gamma_i$. By Lemma 29,

$$R_{\mathbb{F}}^{rc}(A_t, n_t^{1-\varepsilon\gamma/4}) \leq n_t^{4\varepsilon} \quad (42)$$

Let $S = \{t \mid n_t \geq n^{\varepsilon/L}\}$ and $N_S = \prod_{t \in S} n_t$. Then, by Lemma 28,

$$R_{\mathbb{F}}^{rc}\left(\bigotimes_{t \in S} A_t, N_S \left(\sum_{t \in S} n_t^{-\varepsilon\gamma/4}\right)\right) \leq N_S^{4\varepsilon}. \quad (43)$$

Hence,

$$R_{\mathbb{F}}^{rc}\left(\bigotimes_{t \in S} A_t, L \cdot N_S n^{-\varepsilon^2\gamma/(4L)}\right) \leq N_S^{4\varepsilon} \quad (44)$$

Observe that, $n/N_S \leq (n^{\varepsilon/L})^L \leq n^\varepsilon$. Thus, $R_{\mathbb{F}}^{rc}\left(\bigotimes_{t \in [L] \setminus S} A_t, 0\right) \leq n^\varepsilon$. Hence, by Lemma 28,

$$R_{\mathbb{F}}^{rc}(M, n^{1-\psi}) \leq n^{5\varepsilon}, \quad (45)$$

as $n^{-\psi} = \log \log d_k \cdot n^{-\varepsilon^2\gamma/(4 \log \log d_k)} = L n^{-\varepsilon^2\gamma/(4L)}$. ◀

Finally, we are ready to prove Theorem 25.

Proof of Theorem 25. Define $b_\varepsilon = 3/\varepsilon$ and $b_* = \max(b, b_\varepsilon)$. Let $\gamma_b = c_0 \frac{1}{b_*^{3/2} \log^3(b_*)}$.

$\frac{\varepsilon^2}{\log^2(1/\varepsilon)}$, where $c_0 > 0$ is the constant given by Theorem 24. We may assume $c_0 < 1$. Define $N_b = b_*^{24/(\varepsilon\gamma_b)}$.

Denote the size of M_i by d_i for $i \in [k]$ and let $S = \{i \in [k] \mid d_i \leq b_*\}$. Let

$$F = \bigotimes_{i \in S} M_i \quad \text{and} \quad H = \bigotimes_{j \in [k] \setminus S} M_j. \quad (46)$$

Denote by N_F and N_H be the orders of F and H , respectively. Then $N_F \cdot N_H = n$. Let $d_{\max} = \max_i d_i$. By Lemma 30,

$$R_{\mathbb{F}}^{rc}(M, N_H^{1-\psi_H}) \leq N_H^{5\varepsilon}, \quad (47)$$

where

$$\psi_H \geq \frac{\varepsilon^2}{4 \log \log d_{\max}} \cdot \frac{12(\log \log d_{\max})^2}{\varepsilon^3 \log d_{\max}} - \frac{\log \log \log d_{\max}}{\log N_H} \geq \frac{2 \log \log n}{\varepsilon \log n}. \quad (48)$$

At the same time, by Theorem 24, if $N_F \geq N_b$, then

$$R_{\mathbb{F}}^{rc} \left(F, N_F^{1-\gamma_b} \right) \leq N_F^{\varepsilon}. \quad (49)$$

Note, since $b_* \geq 3/\varepsilon$, we have

$$\log \log N_b \leq \log(1/\gamma_b) + 2 \log(b_*) \leq 12 \log(b_*) \leq \gamma_b \varepsilon \log N_b / 2. \quad (50)$$

So, in the case $n \geq N_F \geq N_b$ we have $\gamma_b \geq \frac{2 \log \log n}{\varepsilon \log n}$.

If $\min(N_H, N_F) \geq n^{\varepsilon}$ and $N_F \geq N_b$, then, by Lemma 28,

$$\begin{aligned} R_{\mathbb{F}}^{rc} \left(M, n / \log n \right) &\leq R_{\mathbb{F}}^{rc} \left(M, n(n^{-\varepsilon\psi_H} + n^{-\varepsilon\gamma_b}) \right) \leq \\ &\leq R_{\mathbb{F}}^{rc} \left(H \cdot F, n(N_H^{-\psi_H} + N_F^{-\gamma_b}) \right) \leq R_{\mathbb{F}}^{rc} \left(H, N_H^{1-\psi_H} \right) \cdot R_{\mathbb{F}}^{rc} \left(F, N_F^{1-\gamma_b} \right) \leq n^{5\varepsilon}. \end{aligned} \quad (51)$$

If $N_F \leq n^{\varepsilon}$, then $N_H \geq n^{\varepsilon}$, and by Lemma 28,

$$R_{\mathbb{F}}^{rc} \left(M, n / \log n \right) \leq R_{\mathbb{F}}^{rc} \left(M, N_F N_H^{1-\psi_H} \right) \leq N_H^{5\varepsilon} \cdot R_{\mathbb{F}}^{rc} \left(F, 0 \right) \leq N_H^{5\varepsilon} \cdot N_F \leq n^{6\varepsilon}. \quad (52)$$

Similarly, if $N_H \leq n^{\varepsilon}$ and $N_F \geq N_b$, then $R_{\mathbb{F}}^{rc} \left(M, n / \log n \right) \leq n^{6\varepsilon}$.

Finally, if $N_b > N_F \geq n^{\varepsilon}$, then the size of M is bounded by a function of b and ε . ◀

References

- 1 Josh Alman. Kronecker products, low-depth circuits, and matrix rigidity. In *Proc. 53rd STOC*, pages 772–785. ACM Press, 2021. [arXiv:2102.11992](https://arxiv.org/abs/2102.11992). doi:10.1145/3406325.3451008.
- 2 Josh Alman and Ryan Williams. Probabilistic rank and matrix rigidity. In *Proc. 49th STOC*, pages 17:1–17:23. ACM Press, 2017. doi:10.1145/3055399.3055484.
- 3 László Babai and Bohdan Kivva. Matrix rigidity depends on the target field. In *36th Computational Complexity Conf. (CCC'21)*, volume 200, pages 41:1–41:26. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. doi:10.4230/LIPICs.CCC.2021.41.
- 4 Bruno Codenotti, Pavel Pudlák, and Giovanni Resta. Some structural properties of low-rank matrices related to computational complexity. *Theoretical Computer Science*, 235(1):89–107, 2000. doi:10.1016/S0304-3975(99)00185-1.
- 5 Zeev Dvir and Benjamin L. Edelman. Matrix rigidity and the Croot-Lev-Pach lemma. *Theory of Computing*, 15(8):1–7, 2019. doi:10.4086/toc.2019.v015a008.
- 6 Zeev Dvir and Allen Liu. Fourier and circulant matrices are not rigid. *Theory of Computing*, 16(20):1–48, 2020. doi:10.4086/toc.2020.v016a020.
- 7 Oded Goldreich and Avi Wigderson. On the size of depth-three boolean circuits for computing multilinear functions. *Computational Complexity and Property Testing*, pages 41–86, 2020. doi:10.1007/978-3-030-43662-9_6.
- 8 A. Hedayat and Walter Dennis Wallis. Hadamard matrices and their applications. *Annals of Statistics*, 6(6):1184–1238, 1978. doi:10.1214/aos/1176344370.
- 9 Kathy J. Horadam. *Hadamard matrices and their applications*. Princeton university press, 2012. doi:10.1515/9781400842902.
- 10 Satyanarayana V. Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1–2):1–155, 2009. doi:10.1561/04000000011.
- 11 Pavel Pudlák and Petr Savický. Private communication, cited in [Raz89], 1988.
- 12 Alexander Razborov. On Rigid Matrices. Technical report, Steklov Math. Inst., 1989. (In Russian, <http://people.cs.uchicago.edu/~razborov/files/rigid.pdf>).
- 13 Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Math. Found. Comp. Sci. (MFCS'77)*, pages 162–176. Springer, 1977. doi:10.1007/3-540-08353-7_135.

A Tail bounds

In this appendix we review a classical concentration inequality which we apply to the binomial distribution.

► **Theorem 31** (Bernstein inequality). *Let $L \geq 0$ and $\delta > 0$. Let X_1, \dots, X_n be independent random variables satisfying $|X_i - \mathbb{E}(X_i)| \leq L$ for $i \in [n]$. Let $X = \sum_{i=1}^n X_i$. Then we have*

$$\mathbb{P}(X \geq \mathbb{E}(X) + \delta) \leq \exp\left(-\frac{\delta^2/2}{\sum_{i=1}^n \text{Var}(X_i) + L\delta/3}\right).$$

► **Corollary 32.** *Let $d > 1$. Let X_i be i. i. d. $\{0, 1\}$ -valued random variables such that $\mathbb{P}(X_i = 1) = 1 - \mathbb{P}(X_i = 0) = 1/d$ for $i \in [n]$. Let $X = \sum_{i=1}^n X_i$. Let $\delta < 1/d$. Then*

$$\mathbb{P}\left[X \geq \left(\frac{1}{d} + \delta\right)n\right] \leq \exp\left(-\frac{\delta^2 n}{3d}\right) \quad \text{and} \quad \mathbb{P}\left[X \leq \left(\frac{1}{d} - \delta\right)n\right] \leq \exp\left(-\frac{\delta^2 n}{3d}\right).$$

Proof. Note that $\mathbb{E}(X_i) = 1/d$ and $\text{Var}(X_i) = (d-1)/d^2$. The first bound follows from Theorem 31 with $L = (d-1)/d$. The second bound follows from Theorem 31 with $L = (d-1)/d$ by applying it to $Y_i = \mathbb{E}(X_i) - X_i$. ◀

We also use the following elementary bound.

► **Lemma 33.** *Let $0 \leq k \leq n$. Then $\sum_{i=0}^k \binom{n}{i} \leq \left(\frac{en}{k}\right)^k$.*

B Proof of Observation 14

In this appendix we prove Observation 14.

► **Observation 34.** *For any matrix $A \in \mathbb{F}^{d \times d}$ there exists $B \in \mathbb{F}^{(d-1) \times (d-1)}$, vectors $x, y \in \mathbb{F}^d$, $\lambda \in \{0, 1\} \subseteq \mathbb{F}$ and permutation matrices $P_1, P_2 \in \mathbb{F}^{d \times d}$ such that*

$$A = P_1 \cdot G_d(y)^T \cdot \begin{pmatrix} B & 0 \\ 0 & \lambda \end{pmatrix} \cdot G_d(x) \cdot P_2. \quad (53)$$

Proof. We consider two cases. First, assume that A has rank d . In this case, the basis vector e_d can be written as a linear combination of the columns A_i of A . Moreover, by changing the order of columns (by some permutation matrix, say Q_1) we may assume that the coefficient in front of the last column is non-zero. In other words, there exist coefficients μ_1, \dots, μ_d , with $\mu_d \neq 0$ such that

$$e_d = \mu_1(AQ_1)_1 + \mu_2(AQ_1)_2 + \dots + \mu_d(AQ_1)_d.$$

Define $x_i = -\mu_i/\mu_d$ for $i \in [d-1]$ and $x_d = \frac{1}{\mu_d}$. Denote by A' the matrix consisting of the first $d-1$ columns of AQ_1 . Then

$$AQ_1 = \begin{pmatrix} A' & 0_{d-1} \\ & 1 \end{pmatrix} G_d(x). \quad (54)$$

68:18 Improved Upper Bounds for the Rigidity of Kronecker Products

Since A has full rank, the first $d - 1$ rows of A' span \mathbb{F}^{d-1} . So the last row of A' can be written as a linear combination of the first $d - 1$ rows. Hence, for some vector $y \in \mathbb{F}^d$ with $y_d = 1$,

$$(A')^T = (B \ 0_{d-1}) G_d(y) \quad \Rightarrow \quad A Q_1 = G_d(y)^T \cdot \begin{pmatrix} B & 0 \\ 0 & 1 \end{pmatrix} \cdot G_d(x). \quad (55)$$

If A has rank less than d , there exists a column and a row of A that can be expressed as a linear combination of all other columns and rows of A , respectively. By changing the order of rows and columns we may assume that these are d -th column and d -th row. Then, similarly as above, we see that A can be written in the form (53) with $\lambda = 0$. ◀

The Power of One Clean Qubit in Communication Complexity

Hartmut Klauck ✉

Centre for Quantum Technologies, National University of Singapore, Singapore

Debbie Lim ✉

Centre for Quantum Technologies, National University of Singapore, Singapore

Abstract

We study quantum communication protocols, in which the players' storage starts out in a state where one qubit is in a pure state, and all other qubits are totally mixed (i.e. in a random state), and no other storage is available (for messages or internal computations). This restriction on the available quantum memory has been studied extensively in the model of quantum circuits, and it is known that classically simulating quantum circuits operating on such memory is hard when the additive error of the simulation is exponentially small (in the input length), under the assumption that the polynomial hierarchy does not collapse.

We study this setting in communication complexity. The goal is to consider larger additive error for simulation-hardness results, and to not use unproven assumptions.

We define a complexity measure for this model that takes into account that standard error reduction techniques do not work here. We define a clocked and a semi-unclocked model, and describe efficient simulations between those.

We characterize a one-way communication version of the model in terms of weakly unbounded error communication complexity.

Our main result is that there is a quantum protocol using one clean qubit only and using $O(\log n)$ qubits of communication, such that any classical protocol simulating the acceptance behaviour of the quantum protocol within additive error $1/\text{poly}(n)$ needs communication $\Omega(n)$.

We also describe a candidate problem, for which an exponential gap between the one-clean-qubit communication complexity and the randomized communication complexity is likely to hold, and hence a classical simulation of the one-clean-qubit model within *constant* additive error might be hard in communication complexity. We describe a geometrical conjecture that implies the lower bound.

2012 ACM Subject Classification Theory of computation → Communication complexity; Theory of computation → Quantum complexity theory

Keywords and phrases Quantum Complexity Theory, Quantum Communication Complexity, One Clean Qubit Model

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.69

Related Version *Full Version:* <https://arxiv.org/abs/1807.07762>

Funding This work is funded by the Singapore Ministry of Education and by the Singapore National Research Foundation. Also supported by Majulab UMI 3654.

1 Introduction

The computational power of quantum models of computation with different memory restrictions has been studied in order to understand the use of imperfectly implemented qubits. Some possible types of memory restrictions include having only few qubits that are in a pure state plus an abundance of qubits that start in the totally mixed state [12], having memory that starts in an incompressible state that needs to be returned unchanged at the end of the computation, plus some limited auxiliary space available [8], or simply having very little



© Hartmut Klauck and Debbie Lim;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 69; pp. 69:1–69:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

memory for the computation [2, 20, 16, 3]. The underlying idea in these topics is to study the power of models of quantum computing in which the quantum memory is weak, but the control of this memory is good. This is in contrast to the study of models of quantum computation, where the underlying memory is good, but the control is weak, or restricted, such as the Boson-Sampling model [1]. Both are a step towards understanding the power of quantum computing models that are closer to being implementable than the standard circuit model, and eventually to demonstrate quantum supremacy (i.e., to show that for some problem (of possibly small practical interest) quantum computers that can be built outperform classical computers demonstrably).

This paper explores the potential of a model of quantum communication that uses memory containing only a small number of qubits that start in a known pure state, in particular the power of a having only a single clean qubit (plus many qubits that start in the totally mixed state, i.e., start in a random state).

The one-clean-qubit model proposed by Knill and Laflamme [21] is a model of quantum computing where the memory starts in the tensor product of a single qubit in a pure state $|0\rangle$ with the other m qubits that are in the completely-mixed state, with no further storage allowed. This initial state is described by the density matrix $\rho = |0\rangle\langle 0| \otimes \frac{I}{2^m}$.

The model was originally motivated by the nuclear magnetic resonance (NMR) approach to quantum computing, where the initial state may be highly mixed. Quantum circuits operating on such memory are able to perform tasks that look hard classically, such as estimating Jones polynomials, computing Schatten p -norms, spectral density approximation, testing integrability, computation of fidelity decay [21, 34, 9, 30, 31], just to name a few. [12] showed that quantum circuits under the one-clean-qubit restriction cannot be efficiently classically simulated unless the polynomial hierarchy collapses to the second level. In other words, assuming that the polynomial hierarchy does not collapse, polynomial size quantum circuit operating under the one-clean qubits restriction can have acceptance/rejection probabilities such that any classical randomized circuit that has the same acceptance/rejection probabilities up to additive error $1/\exp(n)$ must have superpolynomial size. We note here that we will not consider simulations with multiplicative error in this paper, since those pose a much stronger requirement on the simulation, for instance the simulating algorithm must replicate events of tiny probability with approximately the same probability, and hence such simulations are much less interesting.

In this paper, we study the hardness of simulating the one-clean-qubit model classically in the model of communication complexity. We will consider simulations of the one-clean-qubit model with different amounts of additive errors, namely $\frac{1}{\text{poly}(n)}$ and $O(1)$.

Organization

After some preliminaries in Section 2, in Section 3 we discuss related work. In Section 4, we sketch our results. In Section 5 we develop our model of quantum communication with one clean qubit. We motivate the main complexity measure and introduce the concepts of clocked and semi-unclocked protocols. Section 6 is about our characterization of one-way communication complexity in our model. Section 8 discusses our main result, which concerns the hardness of classically simulating the one-clean-qubit model with additive error.

2 Preliminaries

Communication Complexity

Yao's [38] model of communication complexity consists of two players, Alice and Bob, who are each given private inputs $x \in X$ and $y \in Y$ respectively. In addition, they both know the function f and agree to a certain communication protocol beforehand. The task they wish to perform is to compute $z = f(x, y)$. Having no knowledge of each others' inputs, they have to communicate with each other in order to obtain the result z . Communication complexity asks the question "how much communication is needed to compute $f(x, y)$?", and assumes that the players have unlimited computational power.

For formal definitions regarding standard types of communication protocols see [24], regarding quantum communication complexity see [11]. We will use the following notations:

► **Definition 1.** $Q(f), R(f)$ denote the quantum (without entanglement) and randomized (with public coin) communication complexities of a function f with error $1/3$. A subscript like $Q_\epsilon(f)$ denotes other errors ϵ .

3 Related Work

There has been a lot of research focusing on the hardness of classical simulations of restricted models of quantum computing under certain assumptions [5, 1, 37, 29, 14, 26, 36, 6, 35]. That is to say, a reasonable assumption in complexity theory leads to the impossibility of efficient sampling by a classical computer according to an output probability distribution that can be generated by a quantum computation model. For instance, it is proven that classical simulation with multiplicative error of the IQP model [5] and Boson sampling [1] is hard, unless the polynomial-time hierarchy collapses.

It is interesting to ask if such a result holds for the one-clean-qubit model as well. Over the past few years, the one-clean-qubit model has been shown to be capable of efficiently solving problems where no efficient classical algorithm is known, such as estimating Jones polynomials, computing Schatten p -norms, spectral density approximation, testing integrability and computation of fidelity decay [21, 34, 9, 30, 31]. It has been conjectured that the one-clean-qubit model can be more powerful than classical computing for some problems. However, there has been no proof for such a conjecture. In [27], it is shown that if the output probability distribution of the one-clean-qubit model can be classically efficiently approximated (with at most an exponentially small additive error) then $BQP \subseteq BPP$. Although the belief that $BQP \neq BPP$ is equivalent to that of $P \neq NP$ or that the polynomial hierarchy does not collapse, there is still a good case for it and the assumption is necessary for simulation hardness anyway. Therefore the results in [27] suggest that the one-clean-qubit model is unlikely to be classically efficiently simulatable with exponentially small additive error.

[26] introduced $DQC1_k$, a modified version of the one-clean-qubit model where the workspace starts with one clean qubit and k qubits are measured at the end of the computation. They showed that the $DQC1_k$ model cannot be efficiently classically simulated for $k \geq 3$ (within constant multiplicative error) unless the polynomial hierarchy collapses.

Later on, [26] showed via circuit complexity that the one-clean-qubit model cannot be efficiently classically simulated with $\frac{1}{\exp(n)}$ additive error unless the polynomial hierarchy collapses to the second level.

All existing results regarding the efficient classical simulation of the one-clean-qubit model are conditional (e.g. rely on non-collapse of the polynomial hierarchy) and require simulations to have exponentially small additive error.

We also mention work on classical memory-restricted communication complexity (e.g. [7]) in which some similar issues appear as in this work.

4 Overview of Results

- Definition of a complexity measure for the one-clean-qubit model in communication complexity:
The complexity measure (cost) of a one-clean-qubit protocol is given by $c \cdot \left(\frac{1}{2\epsilon}\right)$, where c is the communication and ϵ is the bias. We define a clocked and a semi-unclocked version.
- Simulation of a clocked k -clean-qubit models using only one-clean qubit is inexpensive:
Such simulations cost $(c + 1) \cdot \left(\frac{2^k}{\epsilon}\right)^2$, where c is the communication and ϵ is the bias.
- The clocked k -clean-qubit model can be simulated by the semi-unclocked one-clean-qubit model:
Such simulations incur a cost of $O(c \log c) \cdot \left(\frac{2^k}{\epsilon}\right)^2$, where c is the communication and ϵ is the bias.
- Upper and lower bounds on the complexity measure of the one-way one-clean-qubit communication complexity model:
The complexity measure of the one-way one-clean-qubit communication complexity model denoted as $Q_{[1]}^{A \rightarrow B}(f)$ is bounded by $2^{\Omega(PP(f)) - O(\log n)} \leq Q_{[1]}^{A \rightarrow B}(f) \leq 2^{O(PP(f))}$. See [17] for the definition of $PP(f)$.
- Classically simulating the one-clean-qubit model with $\frac{1}{poly(n)}$ additive error requires an exponential increase in communication:
We consider the *MIDDLE* problem and give a quantum protocol with one-clean qubit that requires $O(\log n)$ communication while any classical simulation with $\frac{1}{poly(n)}$ additive error requires $\Omega(n)$ communication.
We stress that in previous results about the hardness of simulating the one-clean-qubit model (in circuit complexity) the additive error must be of size at most $1/exp(n)$ for the simulation to be hard, which stems from low probability events being considered that one would never observe realistically. That means that running the one-clean-qubit circuit as an experiment, and observing an outcome that contradicts classicality is an event that happens only with exponentially small probability, and the classical simulation is only hard because of such extremely low probability events. Our result also uses low probability events, but $1/poly(n)$ is much more reasonable, and the events are observable when repeating such a protocol $poly(n)$ times.
- Simulating the one-clean-qubit model with constant additive error:
We consider a problem ABC as a candidate to show that simulating the one-clean-qubit model with *constant* additive error is hard, and construct a quantum protocol that requires $O(\log n)$ communication using one clean qubit for ABC . We conjecture that any classical simulation with constant additive error requires $\Omega(\sqrt{n})$ communication and give a matching upper bound.

Disclaimer: All I 's used in this paper are identity matrices whose dimensions are clear from the context.

5 Communication Complexity of the One-Clean-Qubit Model

5.1 The One-Clean-Qubit Model

► **Definition 2** (*k*-Clean-Qubit Model). *In a k -clean-qubit protocol, all storage initially consists of only k qubits in a clean state $|0\rangle$, while the rest (m qubits) are in the totally mixed state. The players communicate as in a standard quantum protocol. Only at the end of the computation, a single, arbitrary projective measurement (not depending on the inputs) is performed.*

By this definition, all storage in the one-clean-qubit model consists of only one qubit in a clean state $|0\rangle$, while the rest (m qubits) are in the totally mixed state. This can be described by the density matrix

$$\rho = |0\rangle\langle 0| \otimes \frac{I}{2^m}. \quad (1)$$

A protocol in this model for a function f communicates c qubits. Assume the protocol has a bias of ϵ and hence an error of $\frac{1}{2} - \epsilon$. In general, it is not possible to improve the error to, say, $\frac{1}{3}$. Analogous to allowing the computation to be repeated $O(\frac{1}{\epsilon^2})$ times until a correctness probability of at least $\frac{2}{3}$ is achieved, we define the cost of the (unrepeated) protocol to be $c \cdot (\frac{1}{\epsilon})^2$ qubits.

► **Definition 3** ($Q_{[1]}(f)$). *Let \mathcal{P} denote a one-clean-qubit clocked (explained later) protocol for a function $f : X \times Y \rightarrow \{0, 1\}$, such that 0-inputs are accepted with probability at most $p - \epsilon$ and 1-inputs are accepted with probability at least $p + \epsilon$ for some constant $p > 0$ and that uses communication c at most on all inputs. The cost of \mathcal{P} is then c/ϵ^2 .*

We denote the complexity measure of the clocked one-clean-qubit model by $Q_{[1]}(f) = \inf_{\mathcal{P}} \frac{\text{communication}(\mathcal{P})}{\text{bias}(\mathcal{P})^2}$, where the infimum is over all protocols \mathcal{P} for f .

The motivation behind Definition 3 is that it seems unlikely that the success probability can always be amplified arbitrarily. Therefore, we allow the protocol to run with an arbitrarily bad bias but include the cost that it would take to bring this bias up by a standard amplification (repeat the computation $O(\frac{1}{\text{bias}^2})$ times): in the situation described in Definition 3 by a standard Chernoff bound repeating $t = 4/\epsilon^2$ times (and accepting if at least pt runs accepted) would lead to error at most $1/3$.

There is no prior entanglement allowed in this model because the EPR-pairs could be used to create more pure qubits, simply by sending one qubit from one communicating party to another, who can then make the state $|00\rangle$. It is also essential that measurements are performed only at the end of the computation, or a pure state could be obtained by measuring the state (1).

In our paper, we allow arbitrary projective measurements in the one-clean-qubit model. There are papers such as [34] and [27] defining the one-clean-qubit model in a way such that it measures only one qubit at the end of the computation. However, in Theorem 8, we show that there is only negligible difference between these definitions in communication complexity.

5.2 Clocked and Semi-unclocked Models

We consider two types of models: the clocked model and the semi-unclocked model.

► **Definition 4** (Clocked model). *In the clocked model, the message in round i is computed by a unitary that can depend on i . In other words, the protocol knows i without having to store i anywhere. The communication channel of a clocked model is ghosted, i.e. different qubits can be communicated in different rounds.*

Protocols in the clocked model implicitly use a counter to tell the protocol which round it is in. This counter could be considered as extra classical storage, so we define another model that does not allow this. In that model, however, protocols still need to know when to stop, and that in a sense is a counter, just one that cannot be used “inside” the protocol. Since no intermediate measurements are allowed, we simply switch the protocol off after the correct number of rounds, and measure.

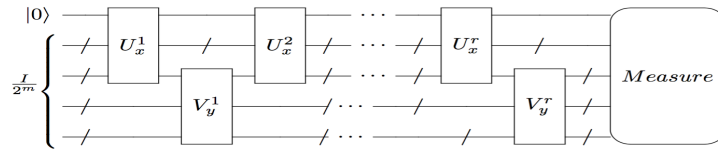


Figure 1 Clocked model.

► **Definition 5 (Semi-unclocked model).** *In the semi-unclocked model, the same unitary must be applied in every round. The protocol terminates after a fixed number of rounds. The communication channel of a semi-unclocked model is fixed, i.e., the same qubits have to be communicated in every round.*

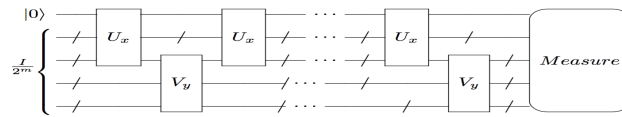


Figure 2 Semi-unclocked model.

► **Example 6.** The inner product modulo 2 problem is defined as $IP_2(x, y) = \sum_i x_i y_i \pmod 2$, where $x, y \in \{0, 1\}^n$. Under the clocked model \hat{P} shown in Figure 3, let U_x^i be Alice’s unitary and let V_y^i be Bob’s for $i = 1 \dots n$. We start with two clean qubits. The first qubit is meant to store Alice’s x_i while the second stores $\sum_i x_i y_i \pmod 2$. The protocol (informally) goes as follows:

In the first round, Alice stores x_1 in the first qubit and sends the two qubits to Bob, who multiplies x_1 in the first qubit with his y_1 and stores the product in the second qubit. He then sends the first qubit back to Alice. For every round $i = 2, \dots, n$,

1. U_x^i first XORs $|x_{i-1}\rangle$ on the first qubit with x_{i-1} , thereby restoring the qubit to $|0\rangle$, before storing the value x_i in it.
2. Alice sends the first qubit to Bob.
3. V_y^i multiplies y_i with x_i (stored in the first qubit) and adds the product to the sum stored in the second qubit modulo 2.
4. Bob sends the first qubit back to Alice.

The communication terminates after a total number of $2n - 1$ rounds and the bias is $\frac{1}{2}$ (i.e. zero error). Bob does the measurement, the total communication is $2n$.

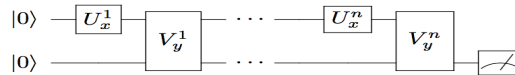


Figure 3 Clocked two-clean-qubit model for computation of inner product modulo 2.

\hat{P} can be simulated with a clocked one-clean-qubit protocol that uses 1 clean qubit and 2 mixed qubits. The unitary \mathcal{M} does the following:

$$\mathcal{M} : \begin{cases} |0\rangle \otimes |0\rangle \otimes |0\rangle \mapsto |1\rangle \otimes |0\rangle \otimes |0\rangle \\ |0\rangle \otimes |z_1\rangle \otimes |z_2\rangle \mapsto |0\rangle \otimes |z_1\rangle \otimes |z_2\rangle \end{cases},$$

where $|z_1\rangle$ or $|z_2\rangle \neq |0\rangle$. Extend to a unitary arbitrarily. In other words, \mathcal{M} flips the first qubit if the next two qubits are both in the $|0\rangle$ state (this happens with probability $\frac{1}{4}$). After applying \mathcal{M} , the protocol is carried out as per \hat{P} . The measurement is done as follows:

- If the first qubit is $|0\rangle$, a “coin toss” is being performed for the output (e.g. measure yet another mixed qubit).
- If the first qubit is $|1\rangle$, the measurement is done as per \hat{P} .

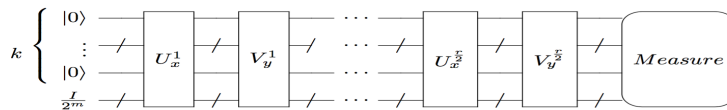
Note that the two measurements can be combined into one.

Therefore, we get an error probability of $\frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8}$, and a bias of $\frac{1}{8}$. The total communication is $2n + 1$ and hence the cost is $64(2n + 1) = O(n)$.

We now compare the k -clean-qubit model with the one-clean-qubit model and also the clocked model with the semi-unclocked model. We prove the following theorems:

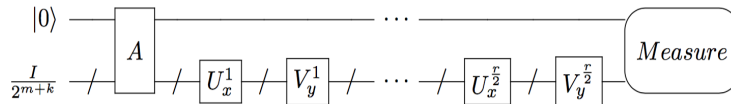
► **Theorem 7.** *Given a clocked k -clean-qubit protocol \mathcal{P} for a function f that has communication c and a bias of ϵ , there exists a clocked one-clean-qubit protocol $\tilde{\mathcal{P}}$ for f that has communication c (or $c + 1$ depending on which player does the measurement), and a bias of $\frac{\epsilon}{2^k}$.*

Proof. The clocked k -clean-qubit protocol \mathcal{P} illustrated in Figure 4 has communication c and a bias of ϵ . Hence, it has an error probability of $\frac{1}{2} - \epsilon$ and cost $\frac{c}{\epsilon^2}$. Denote by U_x^i Alice’s unitaries and by V_y^i Bob’s unitaries for $i = 1, \dots, r$. Note that U_x^i is defined as a unitary on all qubits, but acts only on Alice’s qubits.



■ **Figure 4** Clocked k -clean-qubits protocol \mathcal{P} .

\mathcal{P} can be modified into a clocked one-clean-qubit protocol $\tilde{\mathcal{P}}$ as in Figure 5 with about the same amount of communication.



■ **Figure 5** Clocked one-clean-qubit model $\tilde{\mathcal{P}}$.

In $\tilde{\mathcal{P}}$, the unitary A does a bit flip on the first qubit if the next k qubits are in the $|0\rangle$ state, and does nothing otherwise. All the $k + m$ mixed qubits undergo the same series of unitary transformation as in \mathcal{P} . The measurement in $\tilde{\mathcal{P}}$ is done as follows:

- If the first qubit is $|0\rangle$, a “coin toss” is being done.
- If the first qubit is $|1\rangle$, the measurement is carried out as per \mathcal{P} .

Note that the two measurements can be combined into one.

The communication in $\tilde{\mathcal{P}}$ is c or $c + 1$, depending on which player does the measurement. If the measurement is done by the player who begins the communication, the communication is c . Otherwise, the first qubit has to be sent to the other player for the measurement to be done, causing the communication to be increased to $c + 1$.

The error probability of $\tilde{\mathcal{P}}$ can be computed to be

$$\left(1 - \frac{1}{2^k}\right) \cdot \frac{1}{2} + \frac{1}{2^k} \cdot \left(\frac{1}{2} - \epsilon\right) = \frac{1}{2} - \frac{\epsilon}{2^k}.$$

Hence, the bias decreases from ϵ to $\frac{\epsilon}{2^k}$.

The cost of $\tilde{\mathcal{P}}$ is given by $c \cdot \left(\frac{2^k}{\epsilon}\right)^2$ or $(c + 1) \cdot \left(\frac{2^k}{\epsilon}\right)^2$. ◀

► **Theorem 8.** Given a clocked k -clean-qubit protocol $\tilde{\mathcal{P}}$ for a function $f : X \times Y \rightarrow \{0, 1\}$ with a ghosted communication channel, that does an arbitrary projective measurement with two outcomes, has communication c and a bias of ϵ , there exists a semi-unclocked one-clean-qubit protocol \mathcal{P}_f for f with a fixed communication channel, that does a measurement on one qubit, has communication $O(c \log c)$ and a bias of $\Omega(\frac{\epsilon}{2^k})$.

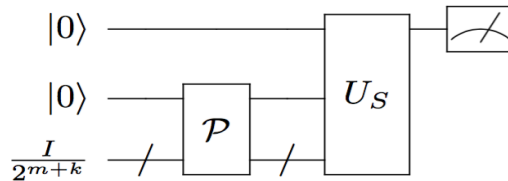
Proof. From Theorem 7, a clocked k -clean qubit protocol $\tilde{\mathcal{P}}$ with a ghosted communication channel that does an arbitrary projective measurement and has communication c and a bias of ϵ , can be modified into a clocked one-clean-qubit protocol \mathcal{P} with a ghosted communication channel, that does an arbitrary projective measurement, has communication $c + 1$ and bias $\frac{\epsilon}{2^k}$. The total number of qubits is $m + k + 1$, with 1 clean qubit.

We would like to turn \mathcal{P} into a protocol \mathcal{P}' that measures only one qubit in the computational basis. This can be done by adding an extra clean qubit and replacing the measurement in \mathcal{P} with a unitary operator U_S and a measurement that measures the newly added qubit in the standard basis. U_S does the following:

$$U_S : \begin{cases} |a\rangle |b_i\rangle \mapsto |a\rangle |b_i\rangle, \text{ for } b_i \in B \\ |a\rangle |b_i\rangle \mapsto |a \oplus 1\rangle |b_i\rangle, \text{ for } b_i \notin B \end{cases} ,$$

where $a \in \{0, 1\}$ and $B = \{b_1, \dots, b_l\}$ is the basis of the subspace $S \subseteq \mathbb{C}^{m+k+1}$, which is a constituent of the observable used to measure the quantum state in \mathcal{P} .

In other words, U_S flips the first qubit on any basis vector $b_i \notin B$, and does nothing otherwise. The resulting protocol \mathcal{P}' is as follows:



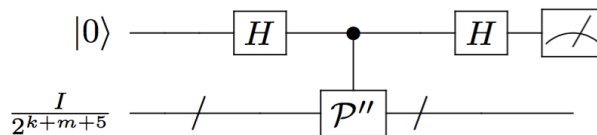
■ **Figure 6** Clocked two-clean-qubit protocol that measures one qubit \mathcal{P}' .

► **Remark 9.** A clocked protocol with a ghosted communication channel can be easily converted to one with fixed channel in which Alice and Bob take turns to send one qubit each. This at most doubles the communication.

In the new protocol, the communication channel is fixed, the total communication is increased to at most $2(c + 1)$, and the bias remains unchanged.

According to [34], the probability of measuring 0 (which corresponds to acceptance) can be made to depend only on the trace of a unitary operator as shown below.

Consider the following trace estimation protocol \mathcal{P}_{main} illustrated in Figure 7,

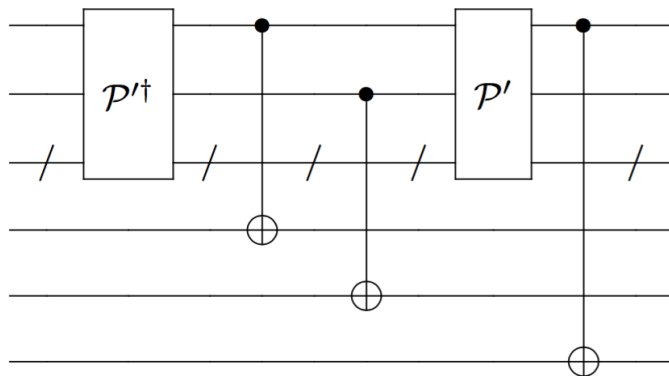


■ **Figure 7** Trace estimation protocol \mathcal{P}_{main} .

which contains the unitary operator \mathcal{P}'' shown in Figure 8. \mathcal{P}_{main} accepts with probability

$$\frac{1}{2} + \frac{\text{Re}(\text{Tr}(\mathcal{P}''))}{2^{d+1}},$$

where $d = m + k + 5$ is the number of qubits in \mathcal{P}'' and $\text{Re}(x)$ is the real part of x .



■ **Figure 8** \mathcal{P}'' .

Let I_ℓ denote the 2^ℓ -dimensional identity matrix. We have that

$$\text{Tr}[(|0\rangle\langle 0| \otimes I_{m+k+1})\mathcal{P}'(|0\rangle^2 \langle 0|^2 \otimes I_{m+k})\mathcal{P}'^\dagger] = \frac{1}{8}\text{Tr}[\mathcal{P}''],$$

because $\text{Tr}[\mathcal{P}''] = \sum_{x \in \{0,1\}^{m+k+5}} \langle x | \mathcal{P}'' | x \rangle$, and so for instance basis vectors $|x\rangle$ that have a 1 in qubit 1 contribute nothing to the sum due to the rightmost CNOT. Similarly, the other CNOTs correspond to the other projection one the left hand side. This equation also shows that the right-hand-side trace is real: up to scaling the left hand side corresponds to a probability of measuring 0 when running \mathcal{P}' on the two-clean-qubit state.

The acceptance probability of \mathcal{P}_{main} is given by

$$\begin{aligned} p_0 &= \frac{1}{2} + \frac{\text{Tr}[\mathcal{P}'']}{2^{k+m+6}} \\ &= \frac{1}{2} + \frac{8 \cdot \text{Tr}[(|0\rangle\langle 0| \otimes I_{m+k+1})\mathcal{P}'(|0\rangle^2 \langle 0|^2 \otimes I_{m+k})\mathcal{P}'^\dagger]}{2^{k+m+6}} \\ &= \frac{1}{2} + \frac{8 \cdot 2^{k+m} \cdot (\frac{1}{2} + \frac{\epsilon}{2^k})}{2^{k+m+6}} \\ &= \frac{1}{2} + \frac{1}{16} + \frac{\epsilon}{2^{k+3}} \end{aligned}$$

► **Remark 10.** The factor of 8 instead of 4 as in [34] is due to the presence of three CNOT gates/extra qubits instead of two.

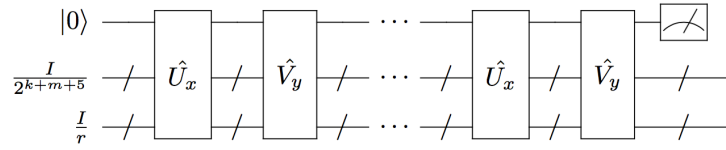
The communication of \mathcal{P}_{main} is four times the communication of \mathcal{P}' , since \mathcal{P}'' runs \mathcal{P}' backwards and forwards, and because the clean control qubit in \mathcal{P}_{main} must be communicated in every round (every round communicates only one qubit in \mathcal{P}'), i.e. the communication becomes $8(c+1)$. The bias decreases to $\frac{\epsilon}{2^{k+3}}$ and is around $\frac{1}{2} + \frac{1}{16}$ instead of $\frac{1}{2}$.

Lastly, we turn \mathcal{P}_{main} into a semi-unclocked protocol \mathcal{P}_f by adding $\log r$ mixed qubits to act as a counter, where r is the number of rounds. The resulting protocol looks as follows:

In \mathcal{P}_f , $\hat{U}_x = (H \otimes I) \cdot U_x \cdot (H \otimes I)$, where

$$U_x : |z\rangle|i\rangle \mapsto (U_x^i |z\rangle) |i\rangle$$

69:10 Communication Complexity with One Clean Qubit



■ **Figure 9** Semi-unlocked one-clean-qubit protocol that measure one qubit \mathcal{P}_f .

and $\hat{V}_y = (H \otimes I) \cdot V_y \cdot (H \otimes I)$, where

$$V_y : |z\rangle |i\rangle \mapsto (V_y^i |z\rangle) |i+1 \pmod r\rangle,$$

for all $z \in \{0, 1\}^{k+m+6}$, for all $i \in \{0, 1\}^{\log r}$ and where U_x^i and V_y^i are the unitaries from \mathcal{P}'' .

This means that, starting from a random j on the counter, the unitaries \hat{V}_y and \hat{U}_x apply V_y^i and U_x^i in the correct, but shifted order. Also note that the Hadamard operators cancel out in between consecutive unitaries, and only the first and last have an effect.

► **Fact 11** (Cyclic property of matrix trace). *The trace of a product of three or more square matrices is invariant under cyclic permutations of the order of multiplication of the matrices.*

Since the acceptance probability of \mathcal{P}_{main} depends only on the trace of the product of the sequence of unitary operators in \mathcal{P}'' , it follows from Fact 11 that the counter can start from any arbitrary $j \pmod r$ without affecting the acceptance probability of \mathcal{P}_f .

The protocol terminates after r rounds of communication. Note that $r = \Theta(c)$, the total communication is now $8(c+1) + O(c \log c) = O(c \log c)$. The bias remains unchanged from \mathcal{P}_{main} , i.e. $\Omega(\frac{\epsilon}{2^k})$. ◀

Applying Theorem 8 to Example 6 gives the following.

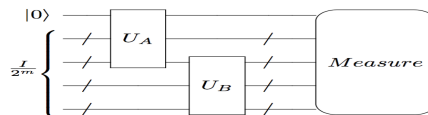
► **Corollary 12.** *The semi-unlocked one-clean-qubit quantum communication complexity of IP_2 is $O(n \log n)$.*

6 One-way Complexity with One Clean Qubit

6.1 The Upper Bound on $Q_{[1]}^{A \rightarrow B}(f)$

Let $Q_{[1]}^{A \rightarrow B}(f)$ denote the complexity measure of a one-way two-player one-clean-qubit protocol. We define a one-way two-player one-clean-qubit protocol as follows:

► **Definition 13** (One-way two-player one-clean-qubit protocol). *The computation in the one-way version of one-clean-qubit protocols starts with a single qubit in the clean state and the rest of the qubits in the totally mixed state. The first player applies her unitary on an arbitrary number of qubits, sends some of the qubits to the next player who also applies his unitary on an arbitrary number of qubits, and does a measurement. The cost is defined as for general one-clean qubit protocols. This can be described by the figure below:*



■ **Figure 10** One-round one-clean-qubit protocol.

Note that this type of protocol is semi-unlocked by definition.

We show an upper bound in terms of the weakly unbounded-error communication complexity.

► **Definition 14** (Weakly unbounded-error protocol, PP). *In a weakly unbounded-error (randomized) protocol (PP protocol), the function f is computed correctly with probability greater than $\frac{1}{2}$ by a classical private coin protocol. The cost of the protocol with a maximum error (over all inputs) of $\frac{1}{2} - \epsilon$ and a maximum communication of c , is given by $PP(f) = c - \lceil \log \epsilon \rceil$. [17]*

We show the following theorem for the upper bound on the communication complexity of the one-clean-qubit one-way protocols (in the appendix):

► **Theorem 15.** $Q_{[1]}^{A \rightarrow B}(f) \leq 2^{O(PP(f))}$.

6.2 The Lower Bound on $Q_{[1]}^{A \rightarrow B}(f)$

► **Theorem 16.** *For all $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ we have $Q_{[1]}^{A \rightarrow B}(f) \geq 2^{\Omega(PP(f)) - O(\log n)}$.*

The proof relies only on the fact that an efficient one-way one-clean-qubit protocol needs to achieve a large enough bias. The communication needed to do so is immaterial for our lower bound, which is quite interesting. In other words, there is a threshold to the bias which simply cannot be passed even if we allow more qubits to be sent. This is in sharp contrast to many common modes of communication with error.

The bound on the achievable bias comes from margin complexity, an important concept in learning theory [25].

Before we delve into the proof we need a few definitions. We define the notion of rectangles and two complexity measures: discrepancy and margin complexity.

► **Definition 17** (Rectangle [24]). *A rectangle in $X \times Y$ is a subset $R \subseteq X \times Y$ such that $R = A \times B$ for some $A \subseteq X$ and $B \subseteq Y$.*

► **Definition 18** (Communication matrix [24]). *The communication matrix M_f of a function $f : X \times Y \rightarrow \{0, 1\}$ is an $(|X| \times |Y|)$ -dimensional matrix, whose rows are indexed by the elements of X and the columns by the elements of Y . The (x, y) entry of M_f is simply defined as $f(x, y)$.*

► **Definition 19** (Discrepancy [24]). *Let $f : X \times Y \rightarrow \{0, 1\}$ be a function, R be any rectangle in the communication matrix, and μ be a probability distribution on $X \times Y$. The discrepancy of f according to μ is*

$$disc_{\mu}(f) = \max_R \left| \Pr_{\mu}[f(x, y) = 0 \text{ and } (x, y) \in R] - \Pr_{\mu}[f(x, y) = 1 \text{ and } (x, y) \in R] \right|.$$

Denote $disc(f) = \min_{\mu} disc_{\mu}(f)$ as the discrepancy of f over all distributions μ on $X \times Y$.

It is known that $PP(f) \geq \Omega(\log(\frac{1}{disc(f)}))$ from Fact 2.8 in [17], and from Theorem 8.1 in [17] we get $PP(f) \leq O(\log(\frac{1}{disc(f)}) + \log n)$.

► **Definition 20** (Margin [25]). *For a function $f : X \times Y \rightarrow \{0, 1\}$, let M_f denote the sign matrix where all entries are $M_f(x, y) = (-1)^{f(x, y)}$. The margin of M_f is given by:*

$$m(M_f) = \sup_{\{a_x\}, \{b_y\}} \min_{x, y} \frac{|\langle a_x | b_y \rangle|}{\|a_x\|_2 \|b_y\|_2},$$

where the supremum is over all systems of vectors (of any length) $\{a_x\}_{x \in X}, \{b_y\}_{y \in Y}$ such that $\text{sign}(\langle a_x | b_y \rangle) = M_f(x, y)$ for all x, y .

69:12 Communication Complexity with One Clean Qubit

The notion of margin complexity determines the extent to which a given class of functions can be learned by large margin classifiers, which is an important class of machine learning algorithms [25].

The proof of Theorem 16 is in the appendix.

7 The Trivial Lower Bound on $Q_{[1]}(f)$

The lower bound on the two-way one-clean-qubit communication complexity $Q_{[1]}(f) \geq \Omega(Q(f))$ is trivial since one-clean-qubit protocols can be turned into standard quantum protocols at their cost. In Appendix C we discuss this lower bound for some well-known functions.

8 Hardness of Classically Simulating the One-Clean-Qubit Model

We now turn to *simulations* of quantum protocols with the one-clean-qubit restriction by randomized protocols. The most demanding definition of simulating a quantum protocol by a randomized protocol is that the randomized protocol must replicate the acceptance probabilities of a given quantum protocol on all inputs, up to some additive error¹.

Our weaker (one-sided) definition of an ϵ -error simulation is:

► **Definition 21** (ϵ -error simulation of a quantum protocol). *Given a quantum protocol \mathcal{P} for a function $f : X \times Y \rightarrow \{0, 1\}$ such that for all inputs $(x, y) \in X \times Y$, \mathcal{P} accepts 1-inputs with probability at least α and accepts 0-inputs with probability at most β . A classical simulation of \mathcal{P} with additive error of ϵ is one that accepts 1-inputs with probability at least $\alpha - \epsilon$ and accepts 0-inputs with probability at most $\beta + \epsilon$.*

► Remark 22. The above definition is nontrivial only if $\alpha - \epsilon > \beta + \epsilon$.

8.1 Simulating the One-Clean-Qubit Model with Polynomially Small Additive Error

We show the following lemma (see the appendix):

► **Lemma 23.** *Given any two-round (Alice \rightarrow Bob \rightarrow Alice) k -clean-qubit quantum protocol (with communication $2k$ and where both messages contain only the k clean qubits) for a function f that accepts 0-inputs with probability at most q and accepts 1-inputs with probability at least p , there exists a two-round one-clean qubit protocol (with communication $2k$) for the same function that accepts 0-inputs with probability at most $\frac{q}{2^k}$ and accepts 1-inputs with probability at least $\frac{p}{2^k}$.*

► **Theorem 24.** *In communication complexity, there exists a function $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ and a one-clean-qubit quantum protocol \mathcal{P} with communication $O(\log n)$ such that simulating \mathcal{P} classically with an allowance of $\frac{1}{n^4}$ additive error requires $\Theta(n)$ communication.*

Proof. Consider the function below:

$$MIDDLE(x, y) = 0 \Leftrightarrow \sum_i x_i y_i = \frac{n}{2}, \quad MIDDLE(x, y) = 1 \Leftrightarrow \sum_i x_i y_i \neq \frac{n}{2},$$

¹ We only consider additive error.

where $x, y \in \{0, 1\}^n$. With Lemma 23 in mind, we design a standard quantum protocol first. We would like to compute the state $\frac{1}{\sqrt{n}} \sum_{i=1}^n (-1)^{x_i y_i} |i\rangle$. This can be done by executing the following quantum protocol \mathcal{P} :

1. Alice prepares the state $\frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle |x_i\rangle$ and sends it to Bob.
2. Bob applies his unitary, which maps the state he received from Alice to $\frac{1}{\sqrt{n}} \sum_{i=1}^n (-1)^{x_i y_i} |i\rangle |x_i\rangle$ and sends the result to Alice.
3. Alice XORs the last qubit with x_i and then traces out that qubit to obtain $\frac{1}{\sqrt{n}} \sum_{i=1}^n (-1)^{x_i y_i} |i\rangle$, applies a Hadamard transformation and does a complete measurement in the computational basis. The protocol outputs 1 if it measures the all-zero string and outputs 0 otherwise.

This protocol requires $2 \log n + 2$ communication and uses $\log n + 1$ clean qubits. Finally, we transform the above protocol into a one-clean-qubit protocol according to Lemma 23.

Now we compute the acceptance probabilities of the standard quantum protocol above. Note that $\langle \cdot | \cdot \rangle$ denotes the inner product.

$$\begin{aligned} \langle H(\frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle (-1)^{x_i y_i}) | 00 \dots 0 \rangle &= \langle \frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle (-1)^{x_i y_i} | H(|00 \dots 0\rangle) \rangle \\ &= \langle \frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle (-1)^{x_i y_i} | \frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle \rangle. \end{aligned} \quad (2)$$

For the case where $\langle x, y \rangle = \sum_{i=1}^n x_i y_i = \frac{n}{2}$, we have $\frac{n}{2}$ 0's and $\frac{n}{2}$ 1's among the $x_i y_i$ and hence, (2) for this case equals to zero, which implies that the protocol rejects 0-inputs with certainty.

For the case where $\langle x, y \rangle = \sum_{i=1}^n x_i y_i = \frac{n}{2} + t$, we have $\frac{n}{2} - t$ 0's and $\frac{n}{2} + t$ 1's and hence, the amplitude from (2) is $\frac{1}{\sqrt{n}} \cdot (\frac{n}{2} + t - (\frac{n}{2} - t)) \frac{1}{\sqrt{n}} = \frac{2t}{n}$, which implies an acceptance probability of $(\frac{2t}{n})^2 = \frac{4t^2}{n^2}$.

Notice that the gap between 0- and 1-inputs is $\frac{4t^2}{n^2}$. Now, simulating \mathcal{P} using only one clean qubit does not change the communication but reduces the acceptance probability of 1-inputs from $\frac{4t^2}{n^2}$ to $\frac{2t^2}{n^3}$ and does not change the acceptance probability of 0-inputs. The gap between the acceptance probability of 0-inputs and 1-inputs is now $\frac{2t^2}{n^3} - 0 = \frac{2t^2}{n^3}$.

We will focus on the 1-inputs with $t = -1$.

We then show that classically simulating the one-clean-qubit protocol with $\frac{1}{n^4}$ additive error for the function $MIDDLE(x, y)$ requires $\Omega(n)$ communication. For this, we use Razborov's analysis of the rectangle bound for the Disjointness problem[32] together with a reduction and the fact that the rectangle bound is not sensitive to acceptance probabilities being small. This shows that any classical protocol that simulates the above quantum protocol within additive error $1/n^4$ needs communication $\Omega(n)$. Details are in Appendix E. ◀

8.2 Simulating the One-Clean-Qubit Model with Constant Additive Error

Previous results about the hardness of simulating the one-clean-qubit model (in circuit complexity) require the additive simulation error to be exponentially small. In the previous subsection we have shown that in communication complexity additive error $1/poly(n)$ is already enough to give a separation (which is also not based on unproven assumptions). Here we consider pushing this even further: can the one-clean-qubit model be simulated classically with constant additive error?

69:14 Communication Complexity with One Clean Qubit

Showing hardness of a classical simulation with constant additive error is equivalent to showing a separation between $Q_{[1]}(f)$ and $R(f)$: regarding both complexity measures efficient error reduction is possible². And showing hardness of a simulation of a quantum protocol for f within a small constant error means showing $R(f)$ is large.

The strength of the one-clean-qubit model is trace-estimation. Any communication-like unitary can have its trace estimated by a quantum protocol with only one clean qubit (compare the proof of Theorem 8). So we look for a hard problem along those lines. A two-party one-way quantum protocol is not a good choice, since the trace of the product of unitaries applied by Alice and Bob is a vector inner product and can be estimated well by known randomized protocols with small error, if the gap of acceptance between one-inputs and zero-inputs is large [23].

For technical reasons (cyclic property of matrix trace), looking for the simplest problem that should exhibit a separation we consider the three-player number-in-hand model³.

We conjecture the following:

► **Conjecture 25.** *There exists a function f and a one-clean-qubit quantum protocol \mathcal{P} that computes f exactly with communication $O(\log n)$ such that simulating \mathcal{P} classically with an allowance of constant additive error requires $\Omega(\sqrt{n})$ communication.*

Consider the number-in-hand ABC (promise) problem involving three parties: Alice, Bob and Charlie, who are each given $n \times n$ matrices A , B and C respectively, where $A, B, C \in SO_n$, where SO_n is the special orthogonal group. The ABC problem is described by the following function:

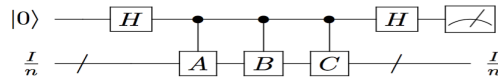
$$ABC(A, B, C) = 1 \iff ABC = I, \quad ABC(A, B, C) = 0 \iff ABC = -I.$$

► **Lemma 26.** *There exists a three-player number-in-hand one-clean-qubit protocol that solves ABC exactly with communication $O(\log n)$.*

Proof. The initial state starts off with one qubit in a pure state $|0\rangle$ and $\log n$ totally mixed qubits. The protocol goes as follows:

1. Alice applies a Hadamard transformation to the clean qubit and obtains $\sigma = H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. She then tensors it with an arbitrary state ρ on $\log n$ qubits (for example $\frac{I}{n}$) and we denote the resulting state as ζ . She then applies her controlled- A unitary to ζ and gets ζ' . Alice send ζ' to Bob.
2. Bob applies his controlled- B unitary to ζ' and gets ζ'' . Bob sends ζ'' to Charlie.
3. Charlie applies his controlled- C unitary to ζ'' and gets ζ''' . He then applies a Hadamard transformation to the first qubit in ζ''' and does a measurement.

The protocol is illustrated in Figure 11.



■ **Figure 11** One-clean-qubit protocol for ABC .

² We defined $Q_{[1]}$ so.

³ In the three-player number-in-hand model, each player sees only their own input.

If $ABC = I$, the composite of controlled A , B and C is the same as that of a controlled-identity unitary, which does nothing to σ . When σ undergoes a Hadamard transformation before being measured, it becomes the $|0\rangle$ state. The protocol outputs 1 if it measures $|0\rangle$. If $ABC = -I$, the composite of controlled A , B and C is similar to that of a controlled- Z unitary, which does a phase flip on $|1\rangle$ in σ , changing it into $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. We denote the phase-flipped σ as σ' . When σ' undergoes a Hadamard transformation before being measured, it becomes the $|1\rangle$ state. The protocol outputs 0 if it measures $|1\rangle$. ◀

Note that the quantum protocol uses the arbitrary state ρ (here $\rho = I/n$) as a catalyst as in [8]. Regarding the randomized complexity of ABC , we prove the following theorem:

► **Theorem 27.** $R(ABC) \leq O(\sqrt{n})$.

See the full version [18] for the proof.

Let us note here that with minor modifications, both the quantum and classical protocols for ABC are one-way and can be run in any order among the players, e.g. Charlie to Alice to Bob or Alice to Charlie to Bob.

It remains an open problem to derive a matching lower bound for the randomized communication complexity of ABC .

► **Conjecture 28.** $R(ABC) \geq \Omega(\sqrt{n})$ as long as n is even.

We now consider a geometric conjecture that implies Conjecture 28. This conjecture says that if we take two sufficiently large subsets of SO_n (the special orthogonal group), choose two operators independently from them, and multiply them, we get something similar to the uniform distribution on all of SO_n .

► **Conjecture 29.** *There are constants $\delta > 0, \gamma > 1$ such that the following is true:*

Let $M, R \subseteq SO_n$ and, for the Haar measure μ on SO_n , let $\mu(M), \mu(R) \geq 2^{-\delta\sqrt{n}}$. Denote by τ the density function of the probability distribution that arises, when $B \in M$ and $C \in R$ are chosen uniformly from these sets independently, and the matrix product BC is formed. Then $\text{Prob}_{A \in SO_n}(\tau(A) \notin [1/\gamma, \gamma]) \leq 2^{-\delta\sqrt{n}}$.

Conjecture 28 follows from Conjecture 29 by an application of the rectangle bound from communication complexity: A large rectangle/box $L \times M \times R$, where $L, M, R \subseteq SO_n$ leads to a τ that is similar to the uniform distribution. Only an exponentially small subset of matrices $A \in SO_n$ has $\tau(A)$ not constant. This also implies that $E_{A \in L} \tau(A) = \Theta(1)$, if we throw out the small subset of $A \in L$ where $\tau(A)$ is too large (this does not affect size or error much.) Denote by β_C the density function of the distribution where a random $B \in M$ is multiplied to a fixed C . $\tau(A) = E_{C \in R} \beta_C(A^*)$.

Define $H = \{(A, B, C) : A, B, C \in SO_n \text{ and } ABC = I\}$ and $G = \{(A, B, C) : A, B, C \in SO_n \text{ and } ABC = -I\}$. It is easy to show that $E_{A \in L} E_{C \in R} [\beta_C(A)] = \frac{\mu(L \times M \times R | H)}{\mu(L \times M \times R)}$. That means that $\mu(L \times M \times R | H)$ and $\mu(L \times M \times R | G)$ differ by at most a constant factor and $L \times M \times R$ has constant error under the distribution that puts weight 1/2 on each of G, H . Hence the rectangle/box $L \times M \times R$ has large error. We use that n is even because otherwise $-I \notin SO_n$. Furthermore in the case of odd n Alice, Bob, and Charlie can simply compute $\det(ABC) = \det(A)\det(B)\det(C)$ in order to determine whether $ABC = I$ or $ABC = -I$. This does not work in the case of even n of course.

We also note that the corresponding conjecture is wrong for O_n , since SO_n is a subgroup of measure 1/2 that serves as a counterexample. Note that SO_n does not have any proper subgroups of size larger than 0.

As weaker conjecture, in which the stated probability is upper bounded by a small constant would be sufficient to give a lower bound on one-way protocols and might be much easier to achieve. We note that in [19] we have recently shown a lower bound for the related aBc problem, in which Alice and Charlie receive vectors from the sphere instead of matrices in a generalized one-way setting. Note that the protocol for Theorem 27 really solves the aBc problem.

9 Conclusion

We investigate a communication complexity model in which all storage consist initially of only one clean qubit plus other qubits that start in the totally mixed state, and where only one projective measurement can be done in the end. Since error reduction is not possible efficiently in this model we define an appropriate complexity measure depending on the bias.

We introduce the notions of clocked protocols with ghosted communication channel and semi-unclocked protocols with fixed communication channel for this model. Efficient simulations of clocked k -clean-qubits protocols by clocked one-clean-qubit protocols as well as simulations of clocked k -clean-qubit protocols by semi-unclocked one-clean-qubit protocols are described. Remarkably, the semi-unclocked model is only less efficient by a logarithmic factor compared to the clocked model.

We study one-way protocols in the model and are able to almost pinpoint their complexity in terms of PP-communication complexity: $2^{\Omega(PP(f)) - O(\log n)} \leq Q_{[1]}^{A \rightarrow B}(f) \leq 2^{O(PP(f))}$, implying that functions when computed using the one-clean-qubit model have a cost of at most $2^{O(m)}$, where m is the input length, and that this is tight for some functions (one-way).

Classically simulating a certain one-clean-qubit protocol for the $MIDDLE(x, y)$ problem with $\frac{1}{\text{poly}(n)}$ additive error is hard, as a classical simulation with such error requires $\Theta(n)$ communication, compared to the $O(\log n)$ communication of the one-clean-qubit protocol.

We conjecture that classically simulating the one-clean-qubit protocol we give for the three-player number-in-hand ABC problem with constant additive error requires $\Omega(\sqrt{n})$ communication, compare to the $O(\log n)$ communication in the one-clean-qubit protocol. We show the corresponding upper bound on $R(ABC)$.

References

- 1 S. Aaronson and A. Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9:143–252, 2013.
- 2 A. Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.
- 3 A. Ambainis, R. Špalek, and R. de Wolf. A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. In *Proceedings of 38th ACM STOC*, pages 618–633, 2006. quant-ph/0511200.
- 4 L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory.pdf. In *Proceedings FOCS*, 1986.
- 5 M. J. Bremner, R. Jozsa, and D. J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. In *Proceedings of the Royal Society A*, volume 467, pages 459–472, 2011.
- 6 D. J. Brod. The complexity of simulating constant-depth boson sampling. *Physical Review A*, 91(4), 2015.
- 7 J. Brody, S. Chen, P. A. Papakonstantinou, H. Song, and X. Sun. Space-bounded communication complexity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 159–172, 2013.

- 8 H. Buhrman, R. Cleve, M. Koucký, B. Loff, and F. Speelman. Computing with a full memory: Catalytic space. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 857–866. ACM, 2014.
- 9 C. Cade and A. Montanaro. The quantum complexity of computing Schatten p -norms, 2017. [arXiv:1706.09279v1](https://arxiv.org/abs/1706.09279v1).
- 10 R. Cleve, W. van Dam, M. Nielsen, and A. Tapp. Quantum entanglement and the communication complexity of the inner product function. In *Proceedings of 1st NASA QCQC conference*, volume 1509 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 1998. [arXiv:quant-ph/9708019](https://arxiv.org/abs/quant-ph/9708019).
- 11 R. de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287:337–353, 2002.
- 12 K. Fujii, H. Kobayashi, T. Morimae, H. Nishimura, S. Tamate, and S. Tani. Power of quantum computation with few clean qubits. *Proceedings of 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, pages 13:1–13:14, 2016.
- 13 R. Jain and H. Klauck. The partition bound for classical communication complexity and query complexity. In *25th Annual Conference on Computational Complexity*, pages 247–258, 2010. [arXiv:0910.4266v2](https://arxiv.org/abs/0910.4266v2).
- 14 R. Jozsa and M. V. den Nest. Classical simulation complexity of extended clifford circuits. *Quantum Information and Computation*, 14(7-8):0633–0648, 2014.
- 15 B. Klartag and O. Regev. Quantum one-way communication is exponentially stronger than classical communication. In *Proceedings of 43rd ACM STOC*, 2011.
- 16 H. Klauck. Quantum and classical communication-space tradeoffs from rectangle bounds. In *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, 24th International Conference*, pages 384–395, 2004.
- 17 H. Klauck. Lower bounds for quantum communication complexity. *SIAM Journal on Computing*, 37(1):20–46, 2007. Earlier version in FOCS'01. [quant-ph/0106160](https://arxiv.org/abs/quant-ph/0106160).
- 18 H. Klauck and D. Lim. The power of one clean qubit in communication complexity. *arXiv*, 2018. [arXiv:1807.07762](https://arxiv.org/abs/1807.07762).
- 19 H. Klauck and D. Lim. The abc problem and equator sampling Renyi divergences. *arXiv*, 2019. [arXiv:1912.11275](https://arxiv.org/abs/1912.11275).
- 20 H. Klauck, R. Špalek, and R. de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM Journal on Computing*, 36(5):1472–1493, 2007. Earlier version in FOCS'04. [quant-ph/0402123](https://arxiv.org/abs/quant-ph/0402123).
- 21 E. Knill and R. Laflamme. On the power of one bit of quantum information. *Phys.Rev.Lett.*, 81:5672–5675, 1998. [arXiv:quant-ph/9802037v1](https://arxiv.org/abs/quant-ph/9802037v1).
- 22 I. Kremer. Quantum communication. Master's Thesis, 1995.
- 23 I. Kremer, N. Nisan, and D. Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999. Earlier version in STOC'95. Correction at <http://www.eng.tau.ac.il/~danar/Public/KNR-fix.ps>.
- 24 E. Kushilevitz and N. Nisan. *Communication Complexity*. CUP, 1997.
- 25 N. Linial and A. Shraibman. Learning complexity vs communication complexity. *Combinatorics, Probability and Computing*, 18:227–245, 2009.
- 26 T. Morimae, K. Fujii, and J. F. Fitzsimons. On the hardness of classically simulating the one clean qubit model. *Phys. Rev. Lett.*, 112, 130502, 2014.
- 27 T. Morimae and T. Koshihara. Classical simulatability of the one clean qubit model, 2014. [arXiv:1405.6840v2](https://arxiv.org/abs/1405.6840v2).
- 28 A. Nayak. Optimal lower bounds for quantum automata and random access codes. In *In Proceedings of 40th IEEE FOCS*, 1999.
- 29 X. Ni and M. V. den Nest. Commuting quantum circuits: Efficient classical simulations versus hardness results. *Quantum Information and Computation*, 13(1-2):0054–0072, 2013.

- 30 D. Poulin, R. Blume-Kohout, R. Laflamme, and H. Ollivier. Exponential speed-up with a single bit of quantum information: Testing the quantum butterfly effect. *Phys. Rev. Lett.*, 92, 177906, 2004.
- 31 D. Poulin, R. Laflamme, G. J. Milburn, and J. P. Paz. Testing integrability with a single bit of quantum information. *Phys. Rev. A*, 68(2):022302–1–022302–6, 2003.
- 32 A. A. Razborov. On the distributional complexity of disjointness. *Theoret. Comput. Sci.*, 106:385–390, 1992.
- 33 A. A. Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya: Mathematics*, 67(1):145, 2003.
- 34 P. W. Shor and S. P. Jordan. Estimating jones polynomials is a complete problem for one clean qubit. *Quantum Information and Computation*, 8:681, 2008. [arXiv:0707.2831v3](https://arxiv.org/abs/0707.2831v3).
- 35 Y. Takahashi, S. Tani, T. Yamazaki, and K. Tanaka. Commuting quantum circuits with few outputs are unlikely to be classically simulatable. *Computing and Combinatorics, 21st International Conference, COCOON 2015*, 9198 of Lecture Notes in Computer Science:223–234, 2015.
- 36 Y. Takahashi, T. Yamazaki, and K. Tanaka. Hardness of classically simulating quantum circuits with unbounded toffoli and fan-out gates. *Quantum Information and Computation*, 14(13-14):1149–1164, 2014.
- 37 B. M. Terhal and D. P. DiVincenzo. Adaptive quantum computation, constant depth quantum circuits and arthur-merlin games. *Quantum Information and Computation*, 4(2):134–145, 2004.
- 38 A. C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing, STOC '79*, pages 209–213. ACM, 1979.

A Open Problems

- Prove Conjecture 28 or the weaker version mentioned above that establishes a lower bound for one-way protocols.
- What are some nontrivial lower bounds on $Q_{[1]}(f)$, for instance what are $Q_{[1]}(DISJ)$ and $Q_{[1]}(ViS)$ (Vector in Subspace [15])? We conjecture that $Q_{[1]}(DISJ) = \Omega(n)$ based on the difficulty of trying to compute the function in the one-clean-qubit model. Suppose that ViS can be computed in the one-clean-qubit communication model efficiently (say with $poly(\log)$ communication), then arbitrary one-way quantum protocols can be simulated with low communication in the one-clean-qubit model. However, we assume that such a supposition seems unlikely and hence we conjecture that $Q_{[1]}(ViS)$ is fairly large, possibly even $Q_{[1]}(ViS) = \Omega(n)$.
- Is $Q_{[1]}(f) > n$ for any function? A candidate for this problem would be a random function chosen from all functions $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$. It would be interesting if the one-clean-qubit model can compute all or most $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ with linear cost.
- What are some examples of functions in which $Q_{[1]}(f) \gg R(f)$ or $Q_{[1]}(f) \ll R(f)$? For instance, for the two-player ABC problem, ABC_2 , described as follows:

$$ABC_2(A_1, A_2, B_1, B_2) = 1 \Leftrightarrow A_1 B_1 A_2 B_2 = I,$$

$$ABC_2(A_1, A_2, B_1, B_2) = 0 \Leftrightarrow A_1 B_1 A_2 B_2 = -I,$$

where A_1, A_2 are Alice's unitaries and B_1, B_2 are Bob's unitaries, $Q_{[1]}(ABC_2) = O(\log n)$. What is $R(ABC_2)$?

- Are there any specific lower bound methods for the semi-unclocked one-clean-qubit protocol?

B Proofs Concerning One-Way Protocols

B.1 Proof of Theorem 15

Consider a c -bit PP -communication protocol \mathcal{P} with bias ϵ where Alice sends a message $T(x)$ of length c to Bob⁴.

1. We define Alice's unitary U_A^x such that
 - If $z = T(x)$, then $U_A^x : |0\rangle|z_1 \cdots z_c\rangle \mapsto |1\rangle|z_1 \cdots z_c\rangle$
 - If $z \neq T(x)$, then $U_A^x : |0\rangle|z_1 \cdots z_c\rangle \mapsto |0\rangle|z_1 \cdots z_c\rangle$
 and extend to a unitary in any possible way, for all $z \in \{0, 1\}^c$. Alice applies U_A^x to the initial state, and computes $U_A^x(|0\rangle\langle 0| \otimes \frac{I}{2^c})U_A^{x\dagger}$.
2. Alice then sends the result σ to Bob. This requires $c + 1$ qubits of communication.
3. Upon receiving σ from Alice, Bob tensors it with $\frac{I}{2}$ and obtains the state $\sigma \otimes \frac{I}{2}$. Bob then applies the unitary V_B^y to the state $\sigma \otimes \frac{I}{2}$, in particular, $V_B^y(\sigma \otimes \frac{I}{2})V_B^{y*}$, as follows

$$V_B^y : \begin{cases} |0\rangle|z_1 \cdots z_{c+1}\rangle \mapsto |0\rangle|z_{c+1}\rangle|z_1 \cdots z_c\rangle \\ |1\rangle|z_1 \cdots z_{c+1}\rangle \mapsto |1\rangle U_B^y \otimes I |z_1 \cdots z_{c+1}\rangle, \end{cases}$$

for all $z \in \{0, 1\}^{c+1}$. That is to say, if the first qubit of $\sigma \otimes \frac{I}{2}$ is 1, V_B^y will apply the protocol unitary U_B^y . Otherwise, a “coin toss” is done by flipping the last qubit over to the second position.

4. Lastly, he does the measurement on the second qubit.

The probability of the correct message is $\frac{1}{2^c}$. With a protocol of bias ϵ (and hence and error of $\frac{1}{2} - \epsilon$), the acceptance probability of the message is $\frac{1}{2^c}(\frac{1}{2} + \epsilon)$. On the other hand, the acceptance probability of the message in the “coin toss” is given by $\frac{1}{2}(1 - \frac{1}{2^c})$. Therefore, we have the total acceptance probability:

$$\begin{aligned} (1 - \frac{1}{2^c})\frac{1}{2} + \frac{1}{2^c}(\frac{1}{2} + \epsilon) &= \frac{1}{2} - \frac{1}{2^{c+1}} + \frac{1}{2^{c+1}} + \frac{\epsilon}{2^c} \\ &= \frac{1}{2} + \frac{\epsilon}{2^c} \end{aligned} \quad (3)$$

The total cost of the protocol is bounded as follows:

$$Q_{[1]}^{A \rightarrow B}(f) \leq (c + 1) \cdot \frac{1}{\epsilon'^2} = (c + 1) \cdot 2^{2c} \cdot \frac{1}{\epsilon^2} \leq 2^{2PP(f)} \cdot (PP(f) + 1) \leq 2^{O(PP(f))}, \quad (4)$$

where $\epsilon' = \frac{\epsilon}{2^c}$ from (3).

B.2 Proof of Theorem 16

Proof. Assume that the protocol measures the first qubit in the computational basis (if not, then a similar construction as in Theorem 8 can be used to make this true). The probability of measuring zero is given by $\frac{1}{2} + \frac{\text{tr}(I_A \otimes U_B^y \cdot U_A^x \otimes I_B)}{2^{m+1}}$ [34], where m is the total number of qubits involved and the bias is the term $\frac{\text{tr}(I_A \otimes U_B^y \cdot U_A^x \otimes I_B)}{2^{m+1}}$. Note that I_A and I_B act on the private qubits of Alice and Bob respectively. Let $U_A^x \otimes I_B = A_x$ and $I_A \otimes U_B^y = B_y$, and it follows that

$$\frac{\text{tr}(I_A \otimes U_B^y \cdot U_A^x \otimes I_B)}{2^{m+1}} = \frac{\text{tr}(B_y A_x)}{2^{m+1}} = \frac{\langle b_y | a_x^T \rangle}{2^{m+1}} = \frac{\langle b_y | a_x^T \rangle}{2\|a_x\|_2\|b_y\|_2},$$

⁴ PP -protocols can be assumed to be one-way without loss of generality [4]

69:20 Communication Complexity with One Clean Qubit

where a_x and b_y are the matrices A_x and B_y viewed as vectors, since A_x and B_y are unitary and hence $\|a_x\|_2 = \|b_y\|_2 = 2^{\frac{m}{2}}$. If the protocol has bias ϵ , then $\frac{\langle b_y | a_x^T \rangle}{2^{m+1}} \geq \epsilon$ for $f(x, y) = 1$ and $\frac{\langle b_y | a_x^T \rangle}{2^{m+1}} \leq -\epsilon$ for $f(x, y) = 0$.

► **Remark 30.** The size of the unitary matrices does not matter, which is good, since there can be an arbitrarily number of private qubits used by the players but never communicated.

We know from the above that the best possible bias satisfies $2\epsilon \leq m(f)$. From Theorem 3.1 in [25] which states that $\text{disc}(A) = \Theta(m(A))$, and from Theorem 8.1 in [17], which states that $PP(f) \leq O(-\log \text{disc}(f) + \log n)$ we have

$$Q_{[1]}^{A \rightarrow B}(f) \geq \frac{4}{m^2(f)} \geq 2^{\Omega(PP(f)) - O(\log n)}. \quad \blacktriangleleft$$

► **Remark 31.** This lower bound holds regardless of how much communication is involved: it follows from the fact that one-way one-clean qubit protocols cannot achieve a better bias.

C The Trivial Lower Bound

$Q(f)$ for some functions is given as below [22, 33, 10, 11, 28]:

- The equality function (EQ) defined as

$$EQ(x, y) = 1 \iff x = y \quad , \quad EQ(x, y) = 0 \iff x \neq y,$$

where $x, y \in \{0, 1\}^n$, has $Q(EQ) = \Theta(\log n)$.

Note: No public coin or entanglement.

- The disjointness function (DISJ) defined as

$$DISJ(x, y) = 1 \iff x \cap y = \emptyset \quad , \quad DISJ(x, y) = 0 \iff x \cap y \neq \emptyset,$$

where $x, y \in \{0, 1\}^n$, has $Q(DISJ) = \Theta(\sqrt{n})$.

Note: $\Omega(\sqrt{n}) \leq Q_{[1]}(DISJ) \leq O(n)$.

- The inner product modulo two function (IP_2) defined as

$$IP_2(x, y) = \sum_i x_i y_i \pmod{2},$$

where $x, y \in \{0, 1\}^n$, has $Q(IP_2) = \Theta(n)$.

Note: $Q_{[1]}^{A \rightarrow B}(IP_2) = 2^{\Theta(n)}$ while $Q_{[1]}(IP_2) = \Theta(n)$.

- The vector in subspace function (ViS) defined as

$$ViS(v, H_0) = 1 \iff v \in H_0 \quad , \quad ViS(v, H_0) = 0 \iff v \in H_0^\perp,$$

where $v \in \mathbb{R}^n$ and $H_0 \subseteq \mathbb{R}^n$ is a subspace with dimension $\frac{n}{2}$, has $Q(ViS) = \Theta(\log n)$.

- The index function (INDEX) defined as

$$INDEX(x, i) = x_i,$$

where $x \in \{0, 1\}^n$ and $1 \leq i \leq n$ has $Q(INDEX) = \Theta(\log n)$.

$Q_{[1]}(EQ)$, $Q_{[1]}(ViS)$ and $Q_{[1]}(INDEX)$ are basically unknown: the lower bounds we know are $\Omega(\log n)$, but the upper bounds we have are $O(n)$ for $INDEX$ and EQ , while Theorem 7 implies $Q_{[1]}(ViS) = O(n^2 \log n)$.

D Proof of Lemma 23

In the quantum protocol, Alice prepares the first message $|\phi_x\rangle$ by applying a protocol unitary $W_x^{(1)}$ to the all-zero state on the k clean qubits, and sends it to Bob. Bob then applies the protocol unitary $V_y^{(1)}$ to the message sent by Alice and sends the result back to her. Alice then applies her second unitary $W_x^{(2)}$ and does a measurement. This protocol has communication $2k$ and accepts 0-inputs with probability at most q and accepts 1-inputs with probability at least p , where $p > q$.

Given any state $|\phi_x\rangle$, we can find an orthonormal basis $\beta_x = \{|\beta_1\rangle \cdots |\beta_{2^k}\rangle\}$ that includes $|\phi_x\rangle$ so that $|\phi_x\rangle$ is a member of the basis and $\sum_{i=1}^{2^k} \frac{|\beta_i\rangle\langle\beta_i|}{2^k} = \frac{I}{2^k}$, such that the state $I/2^k$ is the uniform distribution on the elements in the basis. Consider a one-clean-qubit protocol that simulates the above quantum protocol and goes as follows:

1. We define Alice's unitary $W_x'^{(1)}$ such that
 - If $|\beta_i\rangle = |\phi_x\rangle$, then $W_x'^{(1)} : |0\rangle|\beta_i\rangle \mapsto |1\rangle|\beta_i\rangle$
 - If $|\beta_i\rangle \neq |\phi_x\rangle$, then $W_x'^{(1)} : |0\rangle|\beta_i\rangle \mapsto |0\rangle|\beta_i\rangle$
 and extend to a unitary in any possible way. where $|\beta_i\rangle \in \beta_x$. Alice applies $W_x'^{(1)}$ to the initial state, in particular, computes $\sigma_x = W_x'^{(1)}(|0\rangle\langle 0| \otimes \frac{I}{2^k})W_x'^{(1)\dagger}$.
2. Alice then sends the last k qubits to Bob.
3. Bob applies the unitary $V_y^{(1)}$ to the qubits he received from Alice, in particular computes $\sigma_y = I \otimes V_y^{(1)}(\sigma_x)I \otimes V_y^{(1)\dagger}$, where $\dim(I)=2$. Bob sends the qubits back to Alice.
4. Alice applies her unitary $W_x^{(2)}$ (tensored with identity on the first qubit) to σ_y and measures the first qubit. She outputs 0 if she obtains a measurement result of $|0\rangle$. On the other hand, if she obtains a measurement of $|1\rangle$, she proceeds to execute the measurement of the original quantum protocol. In this case, the acceptance probability of 0-inputs is at most $\frac{q}{2^k}$ and the acceptance probability for 1-inputs is at least $\frac{p}{2^k}$. Note that the two measurements can be combined into one.

The simulation of a k -clean-qubit quantum protocol by a one-clean-qubit protocol is shown in Figure 12:

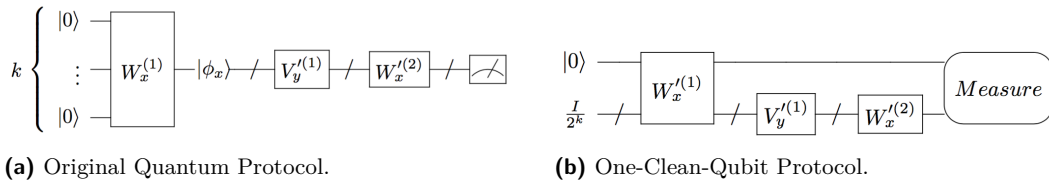


Figure 12 Simulation by a one-clean-qubit protocol.

E The Simulation Lower Bound

First, we insert dummies into the first $\frac{n}{2} - 1$ entries of each string (set all to 1) and the remaining entries are drawn according to a distribution defined in [32].

Consider the linear program (LP) for the rectangle bound (see [13]) as follows, where we set the acceptance probability for 1-inputs to be at least $\alpha = \frac{1}{n^3}$. We consider an additive error of $\frac{1}{n^4}$ and the simulation is required to accept 0-inputs with probability at most $\frac{1}{n^4}$ and accept 1-inputs with probability at least $\frac{2}{n^3} - \frac{1}{n^4} \geq \frac{1}{n^3} = \alpha$. Recall that we consider as 1-inputs only those x, y with $\sum_i x_i y_i = \frac{n}{2} - 1$, and as 0-inputs those with $\sum_i x_i y_i = \frac{n}{2}$.

69:22 Communication Complexity with One Clean Qubit

Denote by \mathcal{R} the set of all rectangles in the communication matrix.

Primal

$$\begin{aligned} & \text{minimize} && \sum_{R \in \mathcal{R}} W_R \\ & \text{subject to} && \sum_{\{R \in \mathcal{R} \mid x, y \in R\}} W_R \geq \alpha, && \text{for all } x, y : \sum_i x_i y_i = \frac{n}{2} - 1 \\ & && \sum_{\{R \in \mathcal{R} \mid x, y \in R\}} -W_R \geq -\frac{1}{n^4}, && \text{for all } x, y : \sum_i x_i y_i = \frac{n}{2} \\ & && W_R \geq 0 \end{aligned}$$

Dual

$$\begin{aligned} & \text{maximize} && \sum_{\{x, y \mid \sum_i x_i y_i = \frac{n}{2} - 1\}} \alpha \gamma_{xy} - \sum_{\{x, y \mid \sum_i x_i y_i = \frac{n}{2}\}} \frac{1}{n^4} \sigma_{xy} \\ & \text{subject to} && \sum_{\{x, y \in R \mid \sum_i x_i y_i = \frac{n}{2} - 1\}} \gamma_{xy} - \sum_{\{x, y \in R \mid \sum_i x_i y_i = \frac{n}{2}\}} \sigma_{xy} \leq 1 && \text{for all } R \in \mathcal{R} \\ & && \sigma_{xy}, \gamma_{xy} \geq 0 \end{aligned}$$

A protocol \mathcal{P} that accepts 1-inputs with probability at least $\frac{1}{n^3}$ and accepts 0-inputs with probability at most $\frac{1}{n^4}$ can be viewed as a probability distribution on deterministic protocols. Each deterministic protocol (in a randomized public-coin protocol) can be represented by a protocol tree. The probabilities of decision trees are given as p_1, p_2, \dots, p_t . Every leaf in each decision tree has an attached rectangle, and a decision: accept or reject. We consider only the rectangles which lead to acceptance, and we assign weight 0 to those rectangles that do not appear in any protocol tree at an accepting leaf and weight $W_R = \sum_{\{i \mid R \text{ accepted in tree } i\}} p_i$

for rectangles appearing in protocol trees i .

▷ **Claim 32.** The constraints in the primal LP hold.

Proof.

- Let (x, y) be a 1-input. Summing up all the probabilities of the decision trees where (x, y) is in a 1-rectangle, we get the LHS of the first inequality constraint, which also corresponds to the acceptance probability, which must exceed α on the RHS.
- Let (x, y) be a 0-input. Adding up the probabilities of decision trees where (x, y) appears in a 1-rectangle will give the LHS of the second inequality constraints, which is at most $1/n^4$ because that is the maximum additive error allowed.
- The nonnegativity constraint is automatically fulfilled since W_R 's are sums of probabilities which must be at least zero. ◁

▷ **Claim 33.** If there is a classical protocol that accepts 1-inputs with probability $\geq \alpha$ and 0-inputs with probability $\leq 1/n^4$ and communication c then there exists a solution of cost 2^c for the primal LP.

Proof. The contribution of each decision tree to W_R is at most $2^c \cdot p_i$, since there are at most 2^c leaves in each decision tree. Therefore,

$$\sum_{R \in \mathcal{R}} W_R \leq \sum_{i=1}^t 2^c \cdot p_i = 2^c. \quad \triangleleft$$

Therefore, in a $\frac{1}{n^4}$ -error simulation of a quantum protocol (that accepts 1-inputs with probability at least $\frac{2}{n^3}$ and accepts 0-inputs with probability 0), the simulating randomized protocol (with communication c) must accept 1-inputs with probability at least $\frac{2}{n^3} - \frac{1}{n^4} \geq \frac{1}{n^3}$ and accept 0-inputs with probability at most $\frac{1}{n^4}$, and hence yield a solution to the primal LP of cost at most 2^c . By LP duality the primal and its dual have the same cost, and we want to show the lower bound for the cost. Hence, we work with the dual.

▷ Claim 34. The cost of the LP is $2^{\Omega(n)}$.

See the full version [18] for the proof.

Connecting Constructive Notions of Ordinals in Homotopy Type Theory

Nicolai Kraus  

University of Nottingham, UK

Fredrik Nordvall Forsberg  

University of Strathclyde, UK

Chuangjie Xu  

fortiss GmbH, München, Germany

Abstract

In classical set theory, there are many equivalent ways to introduce ordinals. In a constructive setting, however, the different notions split apart, with different advantages and disadvantages for each. We consider three different notions of ordinals in homotopy type theory, and show how they relate to each other: A notation system based on Cantor normal forms, a refined notion of Brouwer trees (inductively generated by zero, successor and countable limits), and wellfounded extensional orders. For Cantor normal forms, most properties are decidable, whereas for wellfounded extensional transitive orders, most are undecidable. Formulations for Brouwer trees are usually partially decidable. We demonstrate that all three notions have properties expected of ordinals: their order relations, although defined differently in each case, are all extensional and wellfounded, and the usual arithmetic operations can be defined in each case. We connect these notions by constructing structure preserving embeddings of Cantor normal forms into Brouwer trees, and of these in turn into wellfounded extensional orders. We have formalised most of our results in cubical Agda.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases Constructive ordinals, Cantor normal forms, Brouwer trees

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.70

Related Version *Full proofs are available at:* <https://arxiv.org/abs/2104.02549>

Supplementary Material A formalisation is available:

Software (Agda source code): <https://bitbucket.org/nicolaikraus/constructive-ordinals-in-hott/src>; archived at [swh:1:dir:8f62d289d51a35a3f22fab3b9def1e1dbfcaf909](https://swh.1:dir:8f62d289d51a35a3f22fab3b9def1e1dbfcaf909)

Software (Agda as html): <https://cj-xu.github.io/agda/constructive-ordinals-in-hott/index.html>

Funding *Nicolai Kraus:* The Royal Society, grant reference URF\R1\191055.

Fredrik Nordvall Forsberg: UK National Physical Laboratory Measurement Fellowship project “Dependent types for trustworthy tools”.

Chuangjie Xu: The Humboldt Foundation and the LMUexcellent program.

Acknowledgements We thank the participants of the conferences *Developments in Computer Science* and *TYPES*, as well as Helmut Schwichtenberg and Thorsten Altenkirch for fruitful discussions on this work. We are also grateful to the anonymous reviewers, whose remarks helped us improve the paper.

1 Introduction

The use of ordinals is a powerful tool when proving that processes terminate, when justifying induction and recursion [20, 24], or in (meta)mathematics generally. Unfortunately, the standard definition of ordinals is not very well-behaved constructively, meaning that additional work is required before this tool can be deployed in constructive mathematics or program verification tools based on constructive type theory such as Agda [33], Coq [15] or Lean [18]. Constructively, the classical notion of ordinal fragments into a number of inequivalent



© Nicolai Kraus, Fredrik Nordvall Forsberg, and Chuangjie Xu;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 70; pp. 70:1–70:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

definitions, each with pros and cons. For example, “syntactic” ordinal notation systems [10, 36, 38] are popular with proof theorists, as their concrete character typically mean that equality and the order relation on ordinals are decidable. However, truly infinitary operations such as taking the limit of a countable sequence of ordinals are usually not constructible. We will consider a simple ordinal notation system based on Cantor normal forms [32], designed in such a way that there are no “junk” terms not denoting real ordinals.

Another alternative (based on notation systems by Church [14] and Kleene [28]), popular in the functional programming community, is to consider “Brouwer ordinal trees” \mathcal{O} inductively generated by zero, successor and a “supremum” constructor

$$\text{sup} : (\mathbb{N} \rightarrow \mathcal{O}) \rightarrow \mathcal{O}$$

which forms a new tree for every countable sequence of trees [8, 16, 26]. By the inductive nature of the definition, constructions on trees can be carried out by giving one case for zero, one for successors, and one for suprema, just as in the classical theorem of transfinite induction. However calling the constructor sup is wishful thinking; $\text{sup}(s)$ does not faithfully represent the suprema of the sequence s , since we do not have that e.g. $\text{sup}(s_0, s_1, s_2, \dots) = \text{sup}(s_1, s_0, s_2, \dots)$ – each sequence gives rise to a new tree, rather than identifying trees representing the same suprema. We use the notion of higher inductive types [17, 30] from homotopy type theory [40] to remedy the situation and make a type of Brouwer trees which faithfully represents ordinals. Since our ordinals now can be infinitary, we lose decidability of equality and order relations, but we retain the possibility of classifying an ordinal as a zero, a successor or a limit.

One can also consider extensional wellfounded orders, a variation on the classical set-theoretical axioms more suitable for a constructive treatment [39], which was transferred to the setting of homotopy type theory in the HoTT book [40, Chapter 10], and significantly extended by Escardó [23]. One is then forced to give up most notions of decidability – it is not even possible to decide if a given ordinal is zero, a successor or a limit. However many operations can still be defined on such ordinals, and properties such as wellfoundedness can still be proven. This is also the notion of ordinal most closely related to the traditional notion, and thus the most obviously “correct” notion in a classical setting.



All in all, each of these approaches gives quite a different feel to the ordinals they represent: Cantor normal forms emphasise syntactic manipulations, Brouwer trees how every ordinal can be classified as a zero, successor or limit, and extensional wellfounded orders the set theoretic properties of ordinals. As a consequence, each notion of ordinals is typically used in isolation, with no interaction or opportunities to transfer constructions and ideas from one setting to another – e.g., do the arithmetic operations defined on Cantor normal forms obey the same rules as the arithmetic operations defined on Brouwer trees? The goal of this paper is to answer such questions by connecting together the different notions. We do this firstly by introducing an abstract axiomatic framework of what we expect of any notion of ordinal, and explore to what extent the notions above satisfy these axioms, and secondly by constructing faithful embeddings between the notions, which shows that they all represent a correct notion of ordinal from the point of view of classical set theory.

Contributions

- We identify an axiomatic framework for ordinals and ordinal arithmetic that we use to compare the situations above in the setting of homotopy type theory.
- We define arithmetic operations on Cantor normal forms [32] and prove them uniquely correct with respect to our abstract axiomatisation. This notion of correctness has not been verified for Cantor normal forms previously, as far as we know.

- We construct a higher inductive-inductive type of Brouwer trees, and prove that their order is both wellfounded and extensional – properties which do not hold simultaneously for previous definitions of ordinals based on Brouwer trees. Further, we define arithmetic operations, and show that they are uniquely correct.
- We prove that the “set-theoretic” notion of ordinals [40, Section 10.3] satisfies our axiomatisation of addition and multiplication, and give constructive “taboos”, showing that many operations on these ordinals are not possible constructively.
- We relate and connect these different notions of ordinals by constructing order preserving embeddings from more decidable notions into less decidable ones.

Formalisation and Full Proofs

We have formalised the material on Cantor normal forms and Brouwer trees in cubical Agda [43] at <https://cj-xu.github.io/agda/constructive-ordinals-in-hott/>; see also Escardó’s formalisation [23] of many results on “set-theoretic” ordinals in HoTT. We have marked theorems with formalised and partly formalised proofs using the QED symbols  and  respectively; they are also clickable links to the corresponding machine-checked statement. Moreover, pen-and-paper proofs for all our results can be found in the the arXiv version of the paper.

Our formalisation uses the `{-# TERMINATING #-}` pragma to work around one known bug (issue #4725) and one limitation of the termination checker of Agda: recursive calls hidden under a propositional truncation are not seen to be structurally smaller. Such recursive calls when proving a proposition are justified by the eliminator presentation of [21] (although it would be non-trivial to reduce our mutual definitions to eliminators).

2 Underlying Theory and Notation

We work in and assume basic familiarity with homotopy type theory (HoTT), i.e. Martin-Löf type theory extended with higher inductive types and the univalence axiom [40]. The central concept of HoTT is the Martin-Löf identity type, which we write as $a = b$ – we write $a \equiv b$ for definitional equality. We use Agda notation $(x : A) \rightarrow B(x)$ for the type of dependent functions, and write simply $A \rightarrow B$ if B does not depend on $x : A$. If the type in the domain can be inferred from context, we may simply write $\forall x.B(x)$ for $(x : A) \rightarrow B(x)$. Freely occurring variables are assumed to be \forall -quantified.

We denote the type of dependent pairs by $\Sigma(x : A).B(x)$, and its projections by `fst` and `snd`. We write $A \times B$ if B does not depend on $x : A$. We write \mathcal{U} for a universe of types; we assume that we have a cumulative hierarchy $\mathcal{U}_i : \mathcal{U}_{i+1}$ of such universes closed under all type formers, but we will leave universe levels typically ambiguous.

We call a type A a *proposition* if all elements of A are equal, i.e. if $(x : A) \rightarrow (y : A) \rightarrow x = y$ is provable. We write $\mathbf{hProp} = \Sigma(A : \mathcal{U}).\text{isProp}(A)$ for the type of propositions, and we implicitly insert a first projection if necessary, e.g. for $A : \mathbf{hProp}$, we may write $x : A$ rather than $x : \text{fst}(A)$. A type A is a *set*, $A : \mathbf{hSet}$, if $(x = y) : \mathbf{hProp}$ for every $x, y : A$.

By $\exists(x : A).B(x)$, we mean the *propositional truncation* of $\Sigma(x : A).B(x)$, and if $(a, b) : \Sigma(x : A).B(x)$ then $|(a, b)| : \exists(x : A).B(x)$. The elimination rule of $\exists(x : A).B(x)$ only allows to define functions into propositions. By convention, we write $\exists k.P(k)$ for $\exists(k : \mathbb{N}).P(k)$. Finally, we write $A \uplus B$ for the sum type, $\mathbf{0}$ for the empty type, $\mathbf{1}$ for the type with exactly one element `*`, $\mathbf{2}$ for the type with two elements `ff` and `tt`, and $\neg A$ for $A \rightarrow \mathbf{0}$.

The *law of excluded middle* (LEM) says that, for every proposition P , we have $P \uplus \neg P$. Since we explicitly work with constructive notions of ordinals, we do not assume LEM, but rather use it as a *taboo*: a statement is not provable constructively if it implies LEM. Another, weaker, constructive taboo is the *weak limited principle of omniscience* WLPO: It says that any sequence $s : \mathbb{N} \rightarrow \mathbf{2}$ is either constantly ff, or it is not constantly ff.

3 Three Constructions of Types of Ordinals

We consider three concrete notions of ordinals in this paper, together with their order relations $<$ and \leq . The first notion is the one of *Cantor normal forms*, written Cnf , whose order is decidable. The second, written Brw , are *Brouwer Trees*, implemented as a higher inductive-inductive type. Finally, we consider the type Ord of ordinals that were studied in the HoTT book [40], whose order is undecidable, in general. In the current section, we briefly give the three definitions and leave the discussion of results for afterwards.

3.1 Cantor Normal Forms as a Subset of Binary Trees

In classical set theory, every ordinal α can be written uniquely in Cantor normal form

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \dots + \omega^{\beta_n} \text{ with } \beta_1 \geq \beta_2 \geq \dots \geq \beta_n \quad (1)$$

for some natural number n and ordinals β_i . If $\alpha < \varepsilon_0$, then $\beta_i < \alpha$, and we can represent α as a finite binary tree (with a condition) as follows [10, 12, 25, 32]. Let \mathcal{T} be the type of unlabeled binary trees, i.e. the inductive type with suggestively named constructors $0 : \mathcal{T}$ and $\omega^- \mathbin{+} - : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$. Let the relation $<$ be the *lexicographical order*, i.e. generated by the following clauses:

$$\begin{aligned} 0 &< \omega^a \mathbin{+} b \\ a < c &\rightarrow \omega^a \mathbin{+} b < \omega^c \mathbin{+} d \\ b < d &\rightarrow \omega^a \mathbin{+} b < \omega^a \mathbin{+} d. \end{aligned}$$

We have the map $\text{left} : \mathcal{T} \rightarrow \mathcal{T}$ defined by $\text{left}(0) := 0$ and $\text{left}(\omega^a \mathbin{+} b) := a$ which gives us the left subtree (if it exists) of a tree. A tree is a *Cantor normal form* (CNF) if, for every $\omega^s \mathbin{+} t$ that the tree contains, we have $\text{left}(t) \leq s$, where $s \leq t := (s < t) \uplus (s = t)$; this enforces the condition in (1). For instance, both trees $1 := \omega^0 \mathbin{+} 0$ and $\omega := \omega^1 \mathbin{+} 0$ are CNFs. Formally, the predicate isCNF is defined inductively by the two clauses

$$\begin{aligned} &\text{isCNF}(0) \\ &\text{isCNF}(s) \rightarrow \text{isCNF}(t) \rightarrow \text{left}(t) \leq s \rightarrow \text{isCNF}(\omega^s \mathbin{+} t). \end{aligned}$$

We write $\text{Cnf} := \Sigma(t : \mathcal{T}).\text{isCNF}(t)$ for the type of Cantor normal forms. We often omit the proof of $\text{isCNF}(t)$ and call the tree t a CNF if no confusion is caused.

3.2 Brouwer Trees as a Quotient Inductive-Inductive Type

As discussed in the introduction, *Brouwer ordinal trees* (or simply *Brouwer trees*) are in functional programming often inductively generated by the usual constructors of natural numbers (*zero* and *successor*) and a constructor that gives a Brouwer tree for every sequence of Brouwer trees. To state a refined (*correct* in a sense that we will make precise and prove) version, we need the following notions:

Let A be a type and $\prec : A \rightarrow A \rightarrow \mathbf{hProp}$ be a binary relation. If f and g are two sequences $\mathbb{N} \rightarrow A$, we say that f is *simulated by* g , written $f \lesssim g$, if $f \lesssim g := \forall k. \exists n. f(k) \prec g(n)$. We say that f and g are *bisimilar* with respect to \prec , written $f \approx^\prec g$, if we have both $f \lesssim g$ and $g \lesssim f$. A sequence $f : \mathbb{N} \rightarrow A$ is *increasing* with respect to \prec if we have $\forall k. f(k) \prec f(k+1)$. We write $\mathbb{N} \xrightarrow{\prec} A$ for the type of \prec -increasing sequences. Thus an increasing sequence f is a pair $f \equiv (\bar{f}, p)$ with p witnessing that \bar{f} is increasing, but we keep the first projection implicit and write $f(k)$ instead of $\bar{f}(k)$.

Our type of Brouwer trees is a *quotient inductive-inductive type* [2], where we simultaneously construct the type $\mathbf{Brw} : \mathbf{hSet}$ together with a relation $\leq : \mathbf{Brw} \rightarrow \mathbf{Brw} \rightarrow \mathbf{hProp}$. The constructors for \mathbf{Brw} are

```

zero : Brw
succ : Brw → Brw
limit : (ℕ  $\xrightarrow{\prec}$  Brw) → Brw
bisim: (f g : ℕ  $\xrightarrow{\prec}$  Brw) → f  $\approx^\leq$  g → limit f = limit g,

```

where we write $x < y$ for $\text{succ } x \leq y$ in the type of limit . Simulations thus use \leq and the *increasing* predicate uses \prec , as one would expect. The truncation constructor, ensuring that \mathbf{Brw} is a set, is kept implicit in the paper (but is explicit in the Agda formalisation).

The constructors for \leq are the following, where each constructor is implicitly quantified over the variables $x, y, z : \mathbf{Brw}$ and $f : \mathbb{N} \xrightarrow{\prec} \mathbf{Brw}$ that it contains:

```

 $\leq$ -zero      : zero  $\leq$  x
 $\leq$ -trans     : x  $\leq$  y → y  $\leq$  z → x  $\leq$  z
 $\leq$ -succ-mono: x  $\leq$  y → succ x  $\leq$  succ y
 $\leq$ -cocone    : (k : ℕ) → x  $\leq$  f(k) → x  $\leq$  limit f
 $\leq$ -limiting  : (∀k. f(k)  $\leq$  x) → limit f  $\leq$  x

```

The truncation constructor, which ensures that $x \leq y$ is a proposition, is again kept implicit.

We hope that the constructors of \mathbf{Brw} and \leq are self-explanatory. \leq -cocone ensures that $\text{limit } f$ is indeed an upper bound of f , and \leq -limiting witnesses that it is the *least* upper bound or, from a categorical point of view, the (co)limit of f .

By restricting to limits of increasing sequences, we can avoid multiple representations of the same ordinal (as otherwise e.g. $a = \text{limit}(\lambda_. a)$). It is possible to drop this restriction, if one also strengthens the bisim constructor to witness antisymmetry – however we found this version of \mathbf{Brw} significantly harder to work with.

3.3 Extensional Wellfounded Orders

The third notion of ordinals that we consider is the one studied in the HoTT book [40]. This is the notion which is closest to the classical definition of an ordinal as a set with a trichotomous, wellfounded, and transitive order, without a concrete representation. Requiring trichotomy leads to a notion that makes many constructions impossible in a setting where the law of excluded middle is not assumed. Therefore, when working constructively, it is better to replace the axiom of trichotomy by *extensionality*.

Concretely, an ordinal in the sense of [40, Def 10.3.17] is a type¹ X together with a relation $\prec : X \rightarrow X \rightarrow \mathbf{hProp}$ which is *transitive*, *extensional* (any two elements of X with the same predecessors are equal), and *wellfounded* (every element is accessible, where accessibility is the least relation such that x is accessible if every predecessor of x is accessible.) – we will recall the precise definitions in Section 4. We write \mathbf{Ord} for the type of ordinals in this sense. Note the shift of universes that happens here: the type \mathbf{Ord}_i of ordinals with $X : \mathcal{U}_i$ is itself in \mathcal{U}_{i+1} . We are mostly interested in \mathbf{Ord}_0 , but note that \mathbf{Ord}_0 lives in \mathcal{U}_1 , while \mathbf{Cnf} and \mathbf{Brw} both live in \mathcal{U}_0 .

We also have a relation on \mathbf{Ord} itself. Following [40, Def 10.3.11 and Cor 10.3.13], a *simulation* between ordinals (X, \prec_X) and (Y, \prec_Y) is a function $f : X \rightarrow Y$ such that:

(a) f is monotone: $(x_1 \prec_X x_2) \rightarrow (f x_1 \prec_Y f x_2)$; and

(b) for all $x : X$ and $y : Y$, if $y \prec_Y f x$, then we have an $x_0 \prec_X x$ such that $f x_0 = y$.

We write $X \leq Y$ for the type of simulations between (X, \prec_X) and (Y, \prec_Y) . Given an ordinal (X, \prec) and $x : X$, the *initial segment* of elements below x is given as $X_{/x} := \Sigma(y : X). y \prec x$. Following [40, Def 10.3.19], a simulation $f : X \leq Y$ is *bounded* if we have $y : Y$ such that f induces an equivalence $X \simeq Y_{/y}$. We write $X < Y$ for the type of bounded simulations. This completes the definition of \mathbf{Ord} together with type families \leq and $<$.

4 An Abstract Axiomatic Framework for Ordinals

Which properties do we expect a type of ordinals to have? In this section, we go up one level of abstraction. We consider a type A with type families $<$ and $\leq : A \rightarrow A \rightarrow \mathcal{U}$, and discuss the properties that A with $<$ and \leq can have. In Section 3, we introduced each of the types \mathbf{Cnf} , \mathbf{Brw} , and \mathbf{Ord} together with its relations $<$ and \leq . Note that \leq is the reflexive closure of $<$ in the case of \mathbf{Cnf} , but for \mathbf{Brw} and \mathbf{Ord} , this is not constructively provable. In this section, we consider which properties they satisfy.

4.1 General Notions

A is a *set* if it satisfies the principle of unique identity proofs, i.e. if every identity type $a = b$ with $a, b : A$ is a proposition. Similarly, $<$ and \leq are *valued in propositions* if every $a < b$ and $a \leq b$ is a proposition. A relation $<$ is *reflexive* if we have $a < a$, *irreflexive* if it is pointwise not reflexive $\neg(a < a)$, *transitive* if $a < b \rightarrow b < c \rightarrow a < c$, and *antisymmetric* if $a < b \rightarrow b < a \rightarrow a = b$. Further, the relation $<$ is *connex* if $(a < b) \uplus (b < a)$ and *trichotomous* if $(a < b) \uplus (a = b) \uplus (b < a)$.

► **Theorem 1.** *Each of \mathbf{Cnf} , \mathbf{Brw} , and \mathbf{Ord} is a set, and their relations $<$ and \leq are all valued in propositions. In each case, both $<$ and \leq are transitive, $<$ is irreflexive, and \leq is reflexive and antisymmetric. For \mathbf{Cnf} , the relation $<$ is trichotomous and \leq connex; for \mathbf{Ord} , these statements are equivalent to the law of excluded middle. ◀⚙️*

Proving that \leq for \mathbf{Brw} is antisymmetric is challenging because of the path constructors in the inductive-inductive definition of Brouwer trees. Antisymmetry and other technical properties discussed below require us to characterise the relation \leq more explicitly, using an encode-decode argument [29]. By induction on x and y , we define the family \mathbf{Code} such that $(\mathbf{Code} x y) \leftrightarrow (x \leq y)$. The cases for point constructors are unsurprising; for example, we define

¹ Note that [40, Def 10.3.17] asks for X to be a set, but this follows from the rest of the definition and we therefore drop this requirement.

$$\begin{aligned} \text{Code}(\text{limit } f)(\text{succ } y) &::= \forall k. \text{Code}(f k)(\text{succ } y) \\ \text{Code}(\text{limit } f)(\text{limit } g) &::= \forall k. \exists n. \text{Code}(f k)(g n) . \end{aligned}$$

The difficult part is defining `Code` for the path constructor `bisim`. If for example we have $g \approx h$, we need to show that $\text{Code}(\text{limit } f)(\text{limit } g) = \text{Code}(\text{limit } f)(\text{limit } h)$. The core argument is easy; using the bisimulation $g \approx h$, one can translate between indices for g and h with the appropriate properties. However, this example already shows why this becomes tricky: The bisimulation gives us inequalities (\leq), but the translation requires instances of `Code`, which means that $\text{toCode} : (x \leq y) \rightarrow (\text{Code } x y)$ has to be defined *mutually* with `Code`. This is still not sufficient: In total, the mutual higher inductive-inductive construction needs to simultaneously prove and construct `Code`, `toCode`, versions of transitivity and reflexivity of `Code` as well several auxiliary lemmas. The complete definition is presented in the Agda formalisation (file `BrouwerTree.Code`). Once the definition of `Code` is shown correct, many technical properties are simple consequences.

From now on, we will assume that A is a set and that $<$ and \leq are valued in propositions.

4.2 Extensionality and Wellfoundedness

Following [40, Def 10.3.9], we call a relation $<$ *extensional* if, for all $a, b : A$, we have $(\forall c. c < a \leftrightarrow c < b) \rightarrow b = a$, where \leftrightarrow denotes “if and only if” (functions in both directions). Extensionality of $<$ for `Brw` is true, but non-trivial – note that it fails for the “naive” version of `Brw`, where the path constructor `bisim` is missing.

► **Theorem 2.** *For each of `Cnf`, `Brw`, `Ord`, both $<$ and \leq are extensional.* ◀⚙️

We use the inductive definition of accessibility and wellfoundedness (with respect to $<$) by Aczel [1]. Concretely, the type family $\text{acc} : A \rightarrow \mathcal{U}$ is inductively defined by the constructor

$$\text{access} : (a : A) \rightarrow ((b : A) \rightarrow b < a \rightarrow \text{acc}(b)) \rightarrow \text{acc}(a).$$

An element $a : A$ is called *accessible* if $\text{acc}(a)$, and $<$ is *wellfounded* if all elements of A are accessible. It is well known that the following induction principle can be derived from the inductive presentation [40]:

► **Lemma 3 (Transfinite Induction).** *Let $<$ be wellfounded and $P : A \rightarrow \mathcal{U}$ be a type family such that $\forall a. (\forall b < a. P(b)) \rightarrow P(a)$. Then, it follows that $\forall a. P(a)$.* ◀⚙️

In turn, transfinite induction can be used to prove that there is no infinite decreasing sequence if $<$ is wellfounded: $\neg(\Sigma(f : \mathbb{N} \rightarrow A).(i : \mathbb{N}) \rightarrow f(i+1) < f(i))$. A direct corollary is that if $<$ is wellfounded and valued in propositions, then its reflexive closure $(x < y) \uplus (x = y)$ is also valued in propositions, as $b < a$ and $b = a$ are mutually exclusive propositions.

► **Theorem 4.** *For each of `Cnf`, `Brw`, `Ord`, the relation $<$ is wellfounded.* ◀⚙️

The proof for `Brw` again makes crucial use of our encode-decode characterisation of \leq . Whenever $x < \text{limit } f$, we can use the characterisation to find an $n : \mathbb{N}$ such that $x < f(n)$, which allows us to proceed with an inductive proof of wellfoundedness. Note that the results stated so far in particular mean that `Cnf` and `Brw` can be seen as elements of `Ord` themselves.

4.3 Classification as Zero, a Successor, or a Limit

All standard formulations of ordinals allow us to determine a minimal ordinal *zero* and (constructively) calculate the *successor* of an ordinal, but only some allows us to also calculate the *supremum* or *limit* of a collection of ordinals.

4.3.1 Assumptions


We have so far not required a relationship between $<$ and \leq , but we now need to do so in order for the concepts we define to be meaningful. We assume:

- (A1) $<$ is transitive and irreflexive;
- (A2) \leq is reflexive, transitive, and antisymmetric;
- (A3) we have $(<) \subseteq (\leq)$ and $(< \circ \leq) \subseteq (<)$.

The third condition 3 means that $(b < a) \rightarrow (b \leq a)$ and $(c < b) \rightarrow (b \leq a) \rightarrow (c < a)$. The “symmetric” variation

$$(\leq \circ <) \subseteq (<)$$

is true for Cnf and Brw, but for Ord, it is equivalent to the law of excluded middle – hence, we do not assume it. This constructive failure is known, and can be seen as motivation for *plump* ordinals [39, 37]. Of course, the above assumptions are satisfied if \leq is the reflexive closure of $<$, but we again emphasise that this is not necessarily the case.

► **Theorem 5.** *For each of Cnf, Brw, Ord, assumptions 1 to 3 are satisfied.* 


For the remaining concepts, we assume that $<$ and \leq satisfy the discussed assumptions.

4.3.2 Zero and (Strong) Successors

Let a be an element of A . It is *zero*, or *bottom*, if it is at least as small as any other element

$$\text{is-zero}(a) := \forall b. a \leq b, \tag{2}$$


and we say that the triple $(A, <, \leq)$ has a zero if we have an inhabitant of the type $\Sigma(z : A). \text{is-zero}(z)$. Both the types “being a zero” and “having a zero” are propositions.

► **Theorem 6.** *Cnf, Brw, Ord each have a zero.* 

We say that a is a *successor* of b if it is the least element strictly greater² than b :

$$(a \text{ is-suc-of } b) := (b < a) \times \forall x > b. x \geq a.$$

We say that $(A, <, \leq)$ has *successors* if there is a function $s : A \rightarrow A$ which calculates successors, i.e. such that $\forall b. s(b)$ is-suc-of b . “Calculating successors” and “having successors” are propositional properties, i.e. if a function that calculates successors exists, then it is unique. The following statement is simple but useful. Its proof uses assumption 3.

► **Lemma 7.** *Let $s : A \rightarrow A$ be given. The function s calculates successors if and only if $\forall b x. (b < x) \leftrightarrow (s b \leq x)$.* 

² Note that $>$ and \geq are the obvious symmetric notations for $<$, \leq ; they are *not* newly assumed relations.

Dual to “ a is the least element strictly greater than b ” is the statement that “ b is the greatest element strictly below a ”, in which case it is natural to call b the *predecessor* of a . If a is the successor of b and b the predecessor of a , then we call a the *strong successor* of b :

$$a \text{ is-str-suc-of } b := a \text{ is-suc-of } b \times \forall x < a. x \leq b.$$

We say that A has *strong successors* if there is $s : A \rightarrow A$ which calculates strong successors, i.e. such that $\forall b. s(b)$ is-str-suc-of b . The additional information contained in a strong successor play an important role in our technical development. A function $f : A \rightarrow A$ is *<-monotone* or *≤-monotone* if it preserves the respective relation.

► **Theorem 8.** *Each of the three types Cnf, Brw, Ord has strong successors. The successor functions of Cnf and Brw are both <- and ≤-monotone. For the successor function of Ord, either monotonicity property is equivalent to the law of excluded middle.* ◀⚙️

For Cnf, the successor function is given by adding a leaf, for Brw by the constructor with the same name, and for Ord, one forms the coproduct with the unit type.

4.3.3 Suprema and Limits

Finally, we consider *suprema/least upper bounds* of \mathbb{N} -indexed sequences. We say that a is a *supremum* or the *least upper bound* of $f : \mathbb{N} \rightarrow A$, if a is at least as large as every f_i , and if any other x with this property is at least as large as a :

$$(a \text{ is-sup-of } f) := (\forall i. f_i \leq a) \times (\forall x. (\forall i. f_i \leq x) \rightarrow a \leq x).$$

We say that $(A, <, \leq)$ has *suprema* if there is a function $\sqcup : (\mathbb{N} \rightarrow A) \rightarrow A$ which calculates suprema, i.e. such that $(f : \mathbb{N} \rightarrow A) \rightarrow (\sqcup f)$ is-sup-of f . The supremum of a sequence is unique if it exists, i.e. the type of suprema is propositional for a given sequence f . Both the properties “calculating suprema” and “having suprema” are propositions.

Every $a : A$ is trivially the supremum of the sequence constantly a , and therefore, “being a supremum” does not describe the usual notion of *limit ordinals*. One might consider a a *proper* supremum of f if a is pointwise strictly above f , i.e. $\forall i. f_i < a$. This is automatically guaranteed if f is increasing with respect to $<$, and in this case, we call a the *limit* of f :

$$\begin{aligned} _ \text{ is-lim-of } _ : A \rightarrow (\mathbb{N} \xrightarrow{<} A) \rightarrow \mathcal{U} \\ a \text{ is-lim-of } (f, q) := a \text{ is-sup-of } f. \end{aligned}$$

We say that A has *limits* if there is a function $\text{limit} : (\mathbb{N} \xrightarrow{<} A) \rightarrow A$ that calculates limits.

Note that Cnf cannot have limits since one can construct a sequence (see Theorem 22) which comes arbitrarily close to ε_0 . This motivates the restriction to *bounded* sequences, i.e. a sequence f with a $b : \text{Cnf}$ such that $f_i < b$ for all i .

► **Theorem 9.** *Cnf does not have suprema or limits. Brw has limits of increasing sequences by construction. Ord also has limits of increasing sequences, and moreover limits of weakly increasing sequences (i.e. sequences increasing with respect to ≤).*

Assuming the law of exclude middle, Cnf has suprema (and thus limits) of arbitrary bounded sequences. If Cnf has limits of bounded increasing sequences, then the weak limited principle of omniscience (WLPO) is derivable. ◀⚙️

We expect that it is not constructively possible to calculate suprema (or even binary joins) in Brw, as it seems this would make it possible to decide if a limit reaches past $\omega + 1$ or not, which is a constructive taboo.

4.3.4 Classifiability

For classical set-theoretic ordinals, every ordinal is either zero, a successor, or a limit. We say that a notion of ordinals which allows this is has classification. This is very useful, as many theorems that start with “for every ordinal” have proofs that consider the three cases separately. In the same way as not all definitions of ordinals make it possible to calculate limits, only some formulations make it possible to constructively classify any given ordinal. We already defined what it means to be a zero in (2). We now also define what it means for $a : A$ to be a strong successor or a limit:

$$\text{is-str-suc}(a) := \Sigma(b : A).(a \text{ is-str-suc-of } b) \quad \text{is-lim}(a) := \exists f : \mathbb{N} \rightarrow A. a \text{ is-lim-of } f.$$

All of $\text{is-zero}(a)$, $\text{is-str-suc}(a)$ and $\text{is-lim}(a)$ are propositions; note that this is true even though $\text{is-str-suc}(a)$ is defined without a propositional truncation.

► **Lemma 10.** *Any $a : A$ can be at most one out of $\{\text{zero}, \text{strong successor}, \text{limit}\}$, and in a unique way. In other words, the type $\text{is-zero}(a) \uplus \text{is-str-suc}(a) \uplus \text{is-limit}(a)$ is a proposition.*



We say that an element of A is *classifiable* if it is zero or a strong successor or a limit. We say $(A, <, \leq)$ has *classification* if every element of A is classifiable. By Lemma 10, $(A, <, \leq)$ has classification exactly if the type $\text{is-zero}(a) \uplus \text{is-str-suc}(a) \uplus \text{is-limit}(a)$ is contractible.

► **Theorem 11.** *Cnf and Brw have classification. Ord having classification would imply the law of excluded middle.*



Classifiability corresponds to a case distinction, but the useful principle from classical ordinal theory is the related induction principle:

► **Definition 12** (classifiability induction). *We say that $(A, <, \leq)$ satisfies the principle of classifiability induction if the following holds: For every family $P : A \rightarrow \mathbf{hProp}$ such that*

$$\begin{aligned} \text{is-zero}(a) &\rightarrow P(a) \\ (a \text{ is-str-suc-of } b) &\rightarrow P(b) \rightarrow P(a) \\ (a \text{ is-lim-of } f) &\rightarrow (\forall i. P(f_i)) \rightarrow P(a), \end{aligned}$$

we have $\forall a. P(a)$.

Note that classifiability induction does *not* ask for successors or limits to be computable. Using Lemma 10, we get that classifiability induction implies classification. For the reverse, we need a further assumption:

► **Theorem 13.** *Assume $(A, <, \leq)$ has classification and satisfies the principle of transfinite induction. Then $(A, <, \leq)$ satisfies the principle of classifiability induction.*



It is also standard in classical set theory that classifiability induction implies transfinite induction: showing P by transfinite induction corresponds to showing $\forall x < a. P(x)$ by classifiability induction. In our setting, this would require strong additional assumptions, including the assumption that $(x \leq a)$ is equivalent to $(x < a) \uplus (x = a)$, i.e. that \leq is the reflexive closure of $<$. The standard proof works with several strong assumptions of this form, but we do not consider this interesting or useful, and concentrate on the results which work for the weaker assumptions that are satisfied for Brw and Ord (see Section 4.3.1).

► **Theorem 14.** *Cnf and Brw satisfy classifiability induction, while Ord satisfying it again implies excluded middle.*



4.4 Arithmetic

Using the predicates $\text{is-zero}(a)$, a is-suc-of b , and a is-sup-of f , we can define what it means for $(A, <, \leq)$ to have the standard arithmetic operations. We still work under the assumptions declared in Section 4.3.1 – in particular, we do not assume that e.g. limits can be calculated, which is important to make the theory applicable to Cnf .

► **Definition 15** (having addition). *We say that $(A, <, \leq)$ has addition if there is a function $+ : A \rightarrow A \rightarrow A$ which satisfies the following properties:*

$$\begin{aligned} \text{is-zero}(a) &\rightarrow c + a = c \\ a \text{ is-suc-of } b &\rightarrow d \text{ is-suc-of } (c + b) \rightarrow c + a = d \\ a \text{ is-lim-of } f &\rightarrow b \text{ is-sup-of } (\lambda i. c + f_i) \rightarrow c + a = b \end{aligned} \tag{3}$$

We say that A has unique addition if there is exactly one function $+$ with these properties.

Note that (3) makes an assumption only for (strictly) *increasing* sequences f ; this suffices to define a well-behaved notion of addition, and it is not necessary to include a similar requirement for arbitrary sequences. Since $(\lambda i. c + f_i)$ is a priori not necessarily increasing, the middle term of (3) has to talk about the supremum, not the limit.

Completely analogously to addition, we can formulate multiplication and exponentiation, again without assuming that successors or limits can be calculated:

► **Definition 16** (having multiplication). *Assuming that A has addition, we say that it has multiplication if we have a function $\cdot : A \rightarrow A \rightarrow A$ that satisfies the following properties:*


$$\begin{aligned} \text{is-zero}(a) &\rightarrow c \cdot a = a \\ a \text{ is-suc-of } b &\rightarrow c \cdot a = c \cdot b + c \\ a \text{ is-lim-of } f &\rightarrow b \text{ is-sup-of } (\lambda i. c \cdot f_i) \rightarrow c \cdot a = b \end{aligned}$$

A has unique multiplication if it has unique addition and there is exactly one function \cdot with the above properties.

► **Definition 17** (having exponentiation). *Assume A has addition and multiplication. We say that A has exponentiation with base c if we have a function $\text{exp}(c, -) : A \rightarrow A$ that satisfies the following properties:*

$$\begin{aligned} \text{is-zero}(b) &\rightarrow a \text{ is-suc-of } b \rightarrow \text{exp}(c, b) = a \\ a \text{ is-suc-of } b &\rightarrow \text{exp}(c, a) = \text{exp}(c, b) \cdot c \\ a \text{ is-lim-of } f &\rightarrow \neg \text{is-zero}(c) \rightarrow b \text{ is-sup-of } (\text{exp}(c, f_i)) \rightarrow \text{exp}(c, a) = b \\ a \text{ is-lim-of } f &\rightarrow \text{is-zero}(c) \rightarrow \text{exp}(c, a) = c \end{aligned}$$

A has unique exponentiation with base c if it has unique addition and multiplication, and if $\text{exp}(c, -)$ is unique.

► **Theorem 18.** *Cnf has addition, multiplication, and exponentiation with base ω (all unique), Brw has addition, multiplication and exponentiation with every base (all unique), and Ord has addition and multiplication.* 

70:12 Connecting Constructive Notions of Ordinals

For Cnf, arithmetic is defined by pattern matching on the trees. Addition³ is given as

$$\begin{aligned} 0 + b &::= b \\ a + 0 &::= a \\ (\omega^a + c) + (\omega^b + d) &::= \begin{cases} \omega^b + d & \text{if } a < b \\ \omega^a + (c + (\omega^b + d)) & \text{otherwise,} \end{cases} \end{aligned}$$

multiplication as

$$\begin{aligned} 0 \cdot b &::= 0 \\ a \cdot 0 &::= 0 \\ a \cdot (\omega^0 + d) &::= a + a \cdot d \\ (\omega^a + c) \cdot (\omega^b + d) &::= (\omega^{a+b} + 0) + (\omega^a + c) \cdot d \quad \text{if } b \neq 0, \end{aligned}$$

and exponentiation with base ω by $\omega^a ::= \omega^a + 0$. These definitions are standard. Novel is our proof of correctness in the sense of Definitions 15–17, which we achieve by defining the inverse operations of subtraction and division.

Arithmetic on Brw is defined by recursion on the second argument, following the clauses of the specifications in Definitions 15–17. Since the constructor `limit` only accepts an increasing sequence, it is necessary to prove mutually with the definition that the operations are monotone and preserve increasing sequences. However, the case $c = 0$ needs to be treated separately since neither pointwise multiplication nor exponentiation with 0 preserves increasingness. This makes it crucial to use classification (Theorem 11) and, in particular, that it is decidable whether $c : \text{Brw}$ is zero.

Addition on Ord is given by disjoint union $A \uplus B$ (with $\text{inl}(a) \prec_{A \uplus B} \text{inr}(b)$), and multiplication by Cartesian product $A \times B$ with the reverse lexicographical order. We expect that exponentiation cannot be defined constructively: the “obvious” definition via function spaces gives a wellfounded order assuming the law of excluded middle [27], but it seems unlikely that it can be avoided.


5 Interpretations Between the Notions

In this section, we show how our three notions of ordinals can be connected via structure preserving embeddings.

5.1 From Cantor Normal Forms to Brouwer Trees

The arithmetic operations of Brw allow the construction of a function $\text{CtoB} : \text{Cnf} \rightarrow \text{Brw}$ in a canonical way. We define $\text{CtoB} : \text{Cnf} \rightarrow \text{Brw}$ by:

$$\begin{aligned} \text{CtoB}(0) &::= \text{zero} \\ \text{CtoB}(\omega^a + b) &::= \omega^{\text{CtoB}(a)} + \text{CtoB}(b) \end{aligned}$$

► **Theorem 19.** *The function CtoB preserves and reflects $<$ and \leq , i.e., $a < b \leftrightarrow \text{CtoB}(a) < \text{CtoB}(b)$, and $a \leq b \leftrightarrow \text{CtoB}(a) \leq \text{CtoB}(b)$.* 

³ Caveat: $+$ is a notation for the tree constructor, while $+$ is an operation that we define. We use parenthesis so that all operations can be read with the usual operator precedence.

To show that CtoB preserves $<$, we first prove that Brouwer trees of the form ω^x are additive principal: if $a < \omega^x$ then $a + \omega^x = \omega^x$ – a property not true for the “naive” version of Brouwer trees without path constructors. By reflecting \leq and antisymmetry, we have:

► **Corollary 20.** *The function CtoB is injective.* ◀⚙️

We note that CtoB also preserves all arithmetic operations on Cnf. For multiplication, this relies on $\iota(n) \cdot \omega^x = \omega^x$ for Brw, where $\iota : \mathbb{N} \rightarrow \text{Brw}$ embeds the natural numbers as Brouwer trees, and $\omega := \text{limit } \iota$ – see our formalisation for details.

► **Theorem 21.** *CtoB commutes with addition, multiplication, and exponentiation with base ω .* ◀⚙️

Lastly, as expected, Brouwer trees define bigger ordinals than Cantor normal forms: when embedded into Brw, all Cantor normal forms are below ε_0 , the limit of the increasing sequence $\omega, \omega^\omega, \omega^{\omega^\omega}, \dots$

► **Theorem 22.** *For all $a : \text{Cnf}$, we have $\text{CtoB}(a) < \text{limit } (\lambda k. \omega \uparrow\uparrow k)$, where $\omega \uparrow\uparrow 0 := \omega$ and $\omega \uparrow\uparrow (k + 1) := \omega^{\omega \uparrow\uparrow k}$.* ◀⚙️

5.2 From Brouwer Trees to Extensional Wellfounded Orders

As Brw comes with an order that is extensional, wellfounded, and transitive, it can itself be seen as an element of Ord. Every “subtype” of Brw (constructed by restricting to trees smaller than a given tree) inherits this property, giving a canonical function from Brouwer trees to extensional, wellfounded orders. We define

$$\text{BtoO}(a) = \Sigma(y : \text{Brw}).(y < a).$$

with order relation $(y, p) < (y', p')$ if $y < y'$. This extends to a function $\text{BtoO} : \text{Brw} \rightarrow \text{Ord}$. The first projection gives a simulation $\text{BtoO}(a) \leq \text{Brw}$. Using extensionality of Brw, this implies that BtoO is an embedding from Brw into Ord. Using that $<$ on Brw is propositional, and that carriers of orders are sets, it is also not hard to see that BtoO is order-preserving:

► **Lemma 23.** *The function $\text{BtoO} : \text{Brw} \rightarrow \text{Ord}$ is injective, and preserves $<$ and \leq .* ◀⚙️

A natural question is whether the above result can be strengthened further, i.e. whether BtoO is a simulation. Using LEM to find a minimal simulation witness, this is possible:

► **Theorem 24.** *Under the assumption of the law of excluded middle, the function $\text{BtoO} : \text{Brw} \rightarrow \text{Ord}$ is a simulation.* ◀

We do not know whether the reverse of Theorem 24 is provable, but from the assumption that BtoO is a simulation, we can derive another constructive taboo:

► **Theorem 25.** *If the map $\text{BtoO} : \text{Brw} \rightarrow \text{Ord}$ is a simulation, then WLPO holds.* ◀

We trivially have $\text{BtoO}(\text{zero}) = \mathbf{0}$. One can further prove that BtoO commutes with limits, i.e. $\text{BtoO}(\text{limit}(f)) = \text{lim}(\text{BtoO} \circ f)$. However, BtoO does *not* commute with successors; it is easy to see that $\text{BtoO } x \uplus \mathbf{1} \leq \text{BtoO}(\text{succ } x)$, but the other direction implies WLPO. This also means that BtoO does not preserve the arithmetic operations but “over-approximates” them, i.e. we have $\text{BtoO}(x + y) \geq \text{BtoO } x \uplus \text{BtoO } y$ and $\text{BtoO}(x \cdot y) \geq \text{BtoO } x \times \text{BtoO } y$.

6 Conclusions and Future Directions

We have demonstrated that three very different implementations of ordinal numbers, namely Cantor normal forms (Cnf), Brouwer ordinal trees (Brw), and extensional wellfounded orders (Ord), can be studied in a single abstract setting in the context of homotopy type theory. We hope that our development may shed light on other constructive or formalised approaches to ordinals also in other settings [31, 7, 6, 34].

Cantor normal forms are a formulation where most properties are decidable, while the opposite is the case for extensional wellfounded orders. Brouwer ordinal trees sit in the middle, with some of its properties being decidable. This aspect is not discussed in full in this paper; we only have included Theorem 14. It is easy to see that, for $x : \text{Brw}$, it is decidable whether x is finite; in other words, the predicate $(\omega \leq _) : \text{Brw} \rightarrow \mathbf{hProp}$ is decidable, while $(\omega < _)$ is decidable if and only if WLPO holds.

If x is finite, then the predicates $(x = _)$, $(x \leq _)$, and $(x < _)$ are also decidable. We have a further proof that, if $c : \text{Cnf}$ is smaller than ω^2 , then the families $(\text{CtoB } c \leq _)$ and $(\text{CtoB } c < _)$ are *semidecidable*, where semidecidability can be defined using the *Sierpinski space* [3, 13, 42].

Thus, each of the canonical maps $\text{CtoB} : \text{Cnf} \rightarrow \text{Brw}$ and $\text{BtoO} : \text{Brw} \rightarrow \text{Ord}$ embeds the “more decidable” formulation of ordinals into the “less decidable” one. Naturally, they both also include a “smaller” type of ordinals into a “larger” one: While every element of Cnf represents an ordinal below ϵ_0 , Brw can go much further. It would be interesting to consider more powerful ordinal notation systems such as those based on the Veblen functions [41, 35] or collapsing functions [4, 9], and see how these compare to Brouwer trees. Another avenue for potentially extending Cantor normal forms would be using *superleaves* [19]; we do not know how such a “bigger” version of Cnf would compare to Brw.

Since Brw can be viewed as an element of Ord, the latter can clearly reach larger ordinals than the former. This is of course not surprising; the Burali-Forti argument [5, 11] shows that lower universes cannot reach the same ordinals as higher universes. Another obstruction for Brw to reach the full power of Ord is the fact that Brw only includes limits of \mathbb{N} -indexed sequences. To overcome this problem, one can similarly construct higher number classes as quotient inductive-inductive types, e.g. a type Brw_3 closed under limits of Brw-indexed sequences, and then more generally types Brw_{n+1} closed under limits of Brw_n -indexed sequences, and so on.

Finally, there are interesting connections between the ordinals we can represent and the proof-theoretic strength of the ambient type theory: each proof of wellfoundedness for a system of ordinals is also a lower bound for the strength of the type theory it is constructed in. It is well known that definitional principles such as simultaneous inductive-recursive definitions [22] and higher inductive types [30] can increase the proof-theoretical strength, and so, we hope that they can also be used to faithfully represent even larger ordinals.

References

- 1 Peter Aczel. An introduction to inductive definitions. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 739–782. Elsevier, 1977.
- 2 Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. Quotient inductive-inductive types. In Christel Baier and Ugo Dal Lago, editors, *FoSSaCS '18*, pages 293–310. Springer, 2018. doi:10.1007/978-3-319-89366-2_16.
- 3 Thorsten Altenkirch, Nils Anders Danielsson, and Nicolai Kraus. Partiality, revisited. In Javier Esparza and Andrzej S. Murawski, editors, *FoSSaCS '17*, pages 534–549. Springer, 2017. doi:10.1007/978-3-662-54458-7_31.

- 4 Heinz Bachmann. Die Normalfunktionen und das Problem der ausgezeichneten Folgen von Ordnungszahlen. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich*, 2:115–147, 1950.
- 5 Marc Bezem, Thierry Coquand, Peter Dybjer, and Martín Escardó. The Burali-Forti argument in HoTT/UF in Agda notation, 2020. Available at <https://www.cs.bham.ac.uk/~mhe/TypeTopology/BuraliForti.html>.
- 6 Jasmin Christian Blanchette, Mathias Fleury, and Dmitriy Traytel. Nested multisets, hereditary multisets, and syntactic ordinals in Isabelle/HOL. In Dale Miller, editor, *FSCD '17*, volume 84 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:18, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik.
- 7 Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Cardinals in Isabelle/HOL. In G. Klein and R. Gamboa, editors, *ITP '14*, volume 8558 of *Lecture Notes in Computer Science*, pages 111–127, 2014.
- 8 L. E. J. Brouwer. Zur begründung der intuitionistische mathematik III. *Mathematische Annalen*, 96:451–488, 1926.
- 9 Wilfried Buchholz. A new system of proof-theoretic ordinal functions. *Annals of Pure and Applied Logic*, 32:195–207, 1986. doi:10.1016/0168-0072(86)90052-7.
- 10 Wilfried Buchholz. Notation systems for infinitary derivations. *Archive for Mathematical Logic*, 30:227–296, 1991.
- 11 Cesare Burali-Forti. Una questione sui numeri transfiniti. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 11(1):154–164, 1897.
- 12 Pierre Castéran and Evelyne Contejean. On ordinal notations. Available at <http://coq.inria.fr/V8.2p11/contribs/Cantor.html>, 2006.
- 13 James Chapman, Tarmo Uustalu, and Niccolò Veltri. Quotienting the delay monad by weak bisimilarity. *Mathematical Structures in Computer Science*, 29(1):67–92, 2019.
- 14 Alonzo Church. The constructive second number class. *Bulletin of the American Mathematical Society*, 44:224–232, 1938.
- 15 The Coq Development Team. *The Coq proof assistant reference manual*, 2021. URL: <http://coq.inria.fr>.
- 16 Thierry Coquand, Peter Hancock, and Anton Setzer. Ordinals in type theory. Invited talk at Computer Science Logic (CSL), <http://www.cse.chalmers.se/~coquand/ordinal.ps>, 1997.
- 17 Thierry Coquand, Simon Huber, and Anders Mörtberg. On higher inductive types in cubical type theory. *LICS '18*, 2018.
- 18 Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction – CADE-25 – 25th International Conference on Automated Deduction, Berlin, Germany, August 1–7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer, 2015. doi:10.1007/978-3-319-21401-6_26.
- 19 Nachum Dershowitz. Trees, ordinals and termination. In M. C. Gaudel and J. P. Jouannaud, editors, *TAPSOFT '93*, pages 243–250. Springer, 1993.
- 20 Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.
- 21 Gabe Dijkstra. *Quotient inductive-inductive definitions*. PhD thesis, University of Nottingham, 2017.
- 22 Peter Dybjer and Anton Setzer. Induction-recursion and initial algebras. *Annals of Pure and Applied Logic*, 124(1):1–47, 2003. doi:10.1016/S0168-0072(02)00096-9.
- 23 Martín Escardó. Compact ordinals, discrete ordinals and their relationships, Since 2010–2021. Available at <https://www.cs.bham.ac.uk/~mhe/TypeTopology/Ordinals.html>.
- 24 Robert W. Floyd. Assigning meanings to programs. In J.T. Schwartz, editor, *Symposium on Applied Mathematics*, volume 19, pages 19–32, 1967.
- 25 José Grimm. Implementation of three types of ordinals in Coq. Technical Report RR-8407, INRIA, 2013. Available at <https://hal.inria.fr/hal-00911710>.

- 26 Peter Hancock. *Ordinals and Interactive Programs*. PhD thesis, University of Edinburgh, 2000.
- 27 W Charles Holland, Salma Kuhlmann, and Stephen H McCleary. Lexicographic exponentiation of chains. *Journal of Symbolic Logic*, pages 389–409, 2005.
- 28 S. C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938.
- 29 Daniel Licata and Michael Shulman. Calculating the fundamental group of the circle in homotopy type theory. In *LICS '13*, pages 223–232, 2013. doi:10.1109/LICS.2013.28.
- 30 Peter Lefanu Lumsdaine and Michael Shulman. Semantics of higher inductive types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 169(1):159–208, 2020.
- 31 Panagiotis Manolios and Daron Vroon. Ordinal arithmetic: algorithms and mechanization. *Journal of Automated Reasoning*, 34(4):387–423, 2005.
- 32 Fredrik Nordvall Forsberg, Chuangjie Xu, and Neil Ghani. Three equivalent ordinal notation systems in cubical Agda. In *CPP '20*, pages 172–185. ACM, 2020. doi:10.1145/3372885.3373835.
- 33 Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers University of Technology, 2007.
- 34 Peter H. Schmitt. A mechanizable first-order theory of ordinals. In R. Schmidt and C. Nalon, editors, *TABLEAUX '17*, volume 10501 of *Lecture Notes in Computer Science*, pages 331–346, 2017.
- 35 Kurt Schütte. Kennzeichnung von Ordinalzahlen durch rekursiv erklärte Funktionen. *Mathematische Annalen*, 127:15–32, 1954.
- 36 Kurt Schütte. *Proof Theory*. Springer, 1977.
- 37 Michael Shulman. The surreals contain the plump ordinals. Blog post at <https://homotopytypetheory.org/2014/02/22/surreals-plump-ordinals/>, 2014.
- 38 Gaisi Takeuti. *Proof Theory*. North-Holland, 2 edition, 1987.
- 39 Paul Taylor. Intuitionistic sets and ordinals. *Journal of Symbolic Logic*, 61(3):705–744, 1996.
- 40 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. <http://homotopytypetheory.org/book/>, Institute for Advanced Study, 2013.
- 41 Oswald Veblen. Continuous increasing functions of finite and transfinite ordinals. *Transactions of the American Mathematical Society*, 9(3):280–292, 1908.
- 42 Niccolò Veltri. *A type-theoretical study of nontermination*. PhD thesis, Tallinn University of Technology, 2017.
- 43 Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: a dependently typed programming language with univalence and higher inductive types. In *ICFP '19*, volume 3, pages 87:1–87:29. ACM, 2019.

Maximum Votes Pareto-Efficient Allocations via Swaps on a Social Network

Fu Li ✉

University of Texas at Austin, TX, USA

Xiong Zheng ✉

University of Texas at Austin, TX, USA

Abstract

In recent work, Gourvès, Lesca, and Wilczynski (IJCAI 17) propose a variant of the classic housing markets model in which the matching between agents and objects evolves through Pareto-improving swaps between pairs of agents who are adjacent in a social network. To explore the swap dynamics of their model, they pose several basic questions concerning the set of reachable matchings, and investigate the computational complexity of these questions when the graph structure of the social network is a star, path, or tree, or is unrestricted.

We are interested in how to direct the agents to swap objects with each other in order to arrive at a reachable matching that is both efficient and most agreeable. In particular, we study the computational complexity of reaching a Pareto-efficient matching that maximizes the number of agents who prefer their match to their initial endowments. We consider various graph structures of the social network: path, star, tree, or being unrestricted. Additionally, we consider two assumptions regarding preference relations of agents: strict (ties among objects not allowed) or weak (ties among objects allowed). By designing two polynomial-time algorithms and two NP-hardness reductions, we resolve the complexity of all cases not yet known. Our main contributions include a polynomial-time algorithm for path networks with strict preferences and an NP-hardness result in a star network with weak preferences.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory

Keywords and phrases Housing markets, Distributed process, Algorithms, Complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.71

1 Introduction

Model. Matching indivisible objects to agents is a fundamental problem in both computer science and economics. In a seminal work, Shapley and Scarf [22] introduced the notion of a housing market, which corresponds to the special case of one-sided matching in which there are an equal number of agents and objects, each agent is initially endowed with a distinct object, and each agent is required to be matched to exactly one object. Fruitful applications have arisen from the housing market problem: assigning virtual machines to servers in cloud computers, and allocating graduates to trainee positions. There are two different assumptions regarding preference relations of agents. One is strict, which is a full ordinal list of all objects, and the other one is weak, where agents are allowed to be indifferent between objects. Both preference relations have been widely studied.

However, in practice, assuming that any agent is able to trade with any other agent is a quite strong assumption. Agents tend to trade with agents that they know and trust, and it is hard to let two agents who do not trust each other exchange their objects even if they can mutually get benefits. So Gourvès et al. [13] initiated the line of research on the house market problem where the agents are embedded in a social network. A pair of agents are allowed to swap objects with each other only if (1) both of them will be better off after the swap and (2) they are directly connected (socially tied) via the network. The underlying social network is modeled as an undirected graph. They investigate the swap dynamics of



© Fu Li and Xiong Zheng;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 71; pp. 71:1–71:16

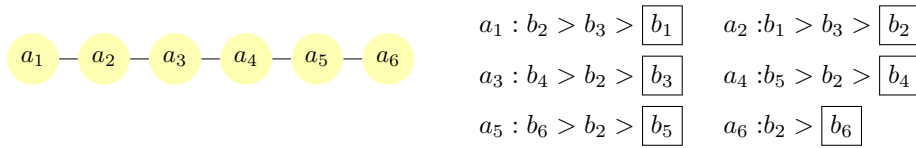
Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the housing market problem in this model by considering the following three computational problems. The first problem, Reachable Object (RO), asks whether there is a sequence of swaps that results in matching a given agent to a given target object. The second problem, Reachable Matching (RM), asks whether there is a sequence of swaps that results in a given target matching. The third problem, Pareto Efficiency (PE), asks how to find a sequence of swaps that results in a Pareto-efficient matching with respect to the set of reachable matchings. Of particular relevance to the present paper is the last problem. We remark that like Gourvès et al., our focus is on Pareto-efficient matchings among reachable matchings (to be formally defined in Section 2). In particular, we focus on reachable matchings that are not Pareto-dominated by a reachable matching, rather than by an arbitrary matching as in the standard meaning. Specifically, we are interested in finding a sequence of swaps that yields a Pareto-efficient matching that maximizes the number of agents who prefer their match to their initial endowments.

Maximum Votes Pareto-Efficient Matching. In housing markets, we typically seek a matching between houses and agents that optimizes some social objects. Pareto efficiency has been widely considered in the context of housing allocation [2, 4, 22], which appears to be the minimal requirement for an allocation to be socially acceptable. However, in the context of social network, Pareto efficiency among reachable matchings is not sufficient to be socially acceptable. In the example of Fig. 1, there is a Pareto-efficient matching that improves only two agents, while there is another Pareto-efficient matching that improves everyone, as shown in Fig. 1.

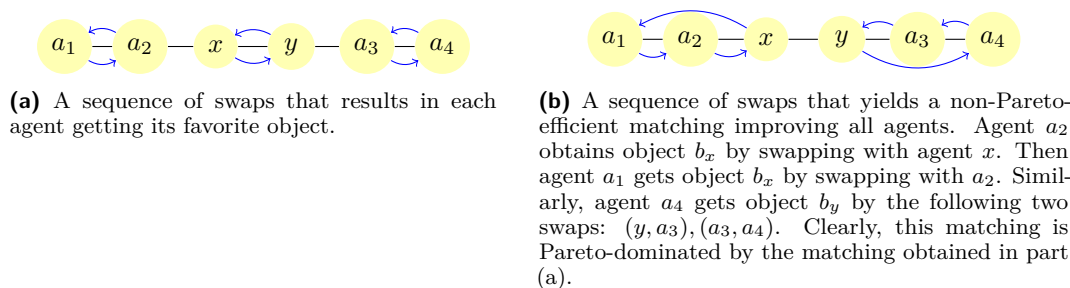


■ **Figure 1** Consider an instance with six agents in $\{a_1, a_2, a_3, a_4, a_5, a_6\}$ and six objects in $\{b_1, b_2, b_3, b_4, b_5, b_6\}$. The boxed objects represent the initial endowments. Initially, only the swaps between agents a_1 and a_2 , and a_2 and a_3 are possible. If we let agents a_1 and a_2 swap their objects first, then we obtain a Pareto-efficient matching in which a_1, a_2 get their best objects. If we let a_2 and a_3 swap their objects first, then additional swaps between pairs (a_1, a_2) and (a_3, a_4) become possible. Indeed, the sequence of swaps, $\{a_2, a_3\}$, $\{a_1, a_2\}$, $\{a_3, a_4\}$, $\{a_4, a_5\}$, and $\{a_5, a_6\}$, yields a different Pareto-efficient matching that improves all agents.

There is a vast literature on refining Pareto-efficiency in object allocation with endowments. Various refining directions are considered, such as fairness ([6, 12]), welfare-maximization ([1, 10]), and stability ([22]). In this work, we propose to refine Pareto-efficiency in the direction of popularity. The notion of popularity was introduced by Abraham et al. [5] for object allocation without endowments. Popularity has gained a lot of attention in the recent past (we refer to the survey by Cseh [11] on this topic). Popular matchings are defined based on comparisons of two matchings with respect to the votes of the agents. Consider an election between two matchings M and M' where vertices are voters. In this M versus M' election, each agent a votes for the matching in $\{M, M'\}$ that agent a prefers, i.e., where agent a gets a better assignment. We say a matching is popular if it never loses a head-to-head election against any matching in the voting instance where matchings are candidates and vertices are voters. However, popular matchings do not always exist: Abraham et al. construct an instance with three agents for which there is no popular matching.

To adapt the popularity notion to housing markets, we introduce the notion of maximum votes matching. We consider the same voting scheme, but focus on the comparison with the initial endowments. Given a reachable matching μ , an agent a votes for the matching if it prefers the object assigned by this matching to its initial object. We refer to the number of agents that prefer μ to the initial matching as the voting number of μ . We define a maximum votes matching to be a reachable matching with the maximum voting number. We remark that maximum votes matchings always exist.

Clearly, not all Pareto efficient matchings have the same voting number. Also, a maximum votes matching is not necessarily a Pareto-efficient matching. Consider the example shown in Fig. 2. Part (a) shows a sequence of swaps that gives a Pareto-efficient matching improving all agents. Part (b) shows a sequence of swaps that also yields a matching that improves all agents, but is not Pareto-efficient. Thus, it is natural to consider the problem of finding a Pareto-efficient matching that has the largest voting number. We refer to this problem as the maximum votes Pareto-efficient (MVPE) matching problem. In particular, the set of MVPE matchings is precisely the intersection of the following two sets: (1) the set of matchings that are Pareto efficient among reachable matchings; (2) the set of reachable matchings with the maximum voting number.



■ **Figure 2** An example in which a non-Pareto-efficient matching improves the most agents. Consider an instance with six agents in $\{a_1, a_2, x, y, a_3, a_4\}$ and six objects in $\{b_1, b_2, b_x, b_y, b_3, b_4\}$. The preferences of the agents and the initial endowments (represented as boxed objects) are given below. The arrows in the figures show where the initial object of each agent goes.

$$\begin{array}{lll}
 a_1 : b_2 > b_x > \boxed{b_1} & x : b_y > b_2 > \boxed{b_x} & a_3 : b_4 > b_y > \boxed{b_3} \\
 a_2 : b_1 > b_x > \boxed{b_2} & y : b_x > b_3 > \boxed{b_y} & a_4 : b_3 > b_y > \boxed{b_4}
 \end{array}$$

Related work. Gourvès et al. [13] study these three problems, RO, RM and PE, in four different graph classes, paths, stars, trees, and general graphs. They only consider strict preferences, and thus they consider twelve problems in total. They study these twelve problems with the goal of either exhibiting a polynomial-time algorithm or establishing NP-completeness. For some of these problems, it is a relatively straightforward exercise to design a polynomial-time algorithm (even for the search version). In particular, this is the case for all three problems on stars, for PE on paths, and for RM on trees (which subsumes RM on paths). Gourvès et al. [13] present an elegant reduction from 2P1N-SAT [23] to establish the NP-completeness of RO on trees. They leverage this result to establish the NP-completeness of RM on general graphs via a reduction from RO on trees. Then, they make use of the latter reduction to establish the NP-hardness of PE on general graphs. The

work of Gourvès et al. [13] left two of the twelve problems open: RO on paths and PE on trees. Subsequently, two sets of authors independently presented polynomial-time algorithms for RO on paths [14, 7]. To the best of our knowledge, PE on trees remains open.

Huang and Xiao [15] study RO and PE with weak preferences. They establish NP-hardness for RO on paths with weak preferences and give a polynomial-time algorithm for RO on stars with weak preferences. Then, they establish NP-hardness for PE on paths with weak preferences via a reduction from RO on paths with weak preferences. They left PE on stars with weak preferences open. Furthermore, they show that finding a reachable matching maximizing total social welfare is NP-hard for a star or a path with weak preferences.

Bentert et al. [7] consider the case where the preference lists have bounded length. Saffidine and Wilczynski [21] propose an alternative version of RO where an agent can be guaranteed a certain level of satisfaction regardless of the actual exchanges. Müller and Bentert [20] study RM on cliques and cycles. Aspects related to social connectivity are also addressed in recent work on envy-free allocations [8, 9] and on a trade-off between efficiency and fairness [16].

There are several works on ensuring popularity. For settings where a popular matching does not exist, Kavitha et al. [18] study how to minimally augment the preferences to guarantee the existence of a popular matching. This problem has been shown to be NP-hard. Another way to ensure popularity is to consider mixed matchings, i.e., lotteries over matchings; and the popularity property is straightforward to carry over; Kavitha et al. [17] show that a popular mixed matching always exists and propose an efficient algorithm to find one. McCutchen [19] proposes a least-unpopularity criterion to find the “most” popular matching; finding this least-unpopular matching is NP-hard.

Our results. Our main contributions are summarized in Table 1. Our main positive result is a polynomial-time algorithm for finding a maximum votes Pareto-efficient matching in a path network with strict preferences. To achieve this result, we first present an algorithm for finding a maximum votes matching by studying the structure of a reachable matching in a path network. Specifically, we show that the improved agents in any reachable matching can be decomposed into a set of agent intervals, where each agent interval is a set of consecutive agents in the path and all agents in this interval can improve their allocation by swapping objects within the interval. Moreover, given a disjoint union of such agent intervals, we can recover a sequence of swaps improving agents in this disjoint union. Thus, we reduce the problem of computing the maximum votes number to the problem of finding the disjoint union of such agent intervals that covers the most agents; polynomial-time algorithms are known for the latter problem.

To find a maximum votes Pareto-efficient matching, observe that there is a maximum votes matching that is also Pareto-efficient. Thus, to find a maximum votes Pareto-efficient matching, it is enough to find a Pareto-efficient matching among maximum votes matchings. To achieve that, we carefully adapt the serial dictatorship algorithm [3] to ensure that each time a dictator agent a is assigned to an object b , where the current partial assignment remains consistent with a maximum votes matching. In particular, we present a subroutine to find the most favorite object b that can be assigned to agent a such that there is a sequence of swaps letting a get b and improving the most agents. In Section 3.4, we use a delicate induction argument to prove the correctness of this subroutine.

In Section 4, we present a simple polynomial-time algorithm for finding a maximum votes Pareto-efficient matching on stars with strict preferences shown

■ **Table 1** Our results and related work. Results in parentheses are implied by other table entries.

	Strict	Weak
Path	poly-time [Section 3]	NP-hard ([15])
Star	poly-time [Section 4]	NP-hard [Section 4]
Tree	NP-hard [Section 5]	(NP-hard)
General	(NP-hard)	(NP-hard)

In terms of negative results, in Section 4, we show that the MVPE problem in star networks with weak preferences is NP-hard by designing a novel reduction from the 3-SAT problem. In Section 5, we prove that the MVPE problem in tree networks with strict preferences is NP-hard by reducing from the RO problem in tree networks with strict preferences, which is known to be NP-complete [13].

Due to space restrictions, some of the proofs are omitted, or are merely sketched. Complete proofs will be provided in the full version of this paper.

2 Preliminaries

We define an object allocation framework (OAF) on a social network as a triple $F = (A, B, \geq, E)$ where A is a set of agents, B is a set of objects such that $|A| = |B|$, \geq is a collection of linear orderings $\{\geq_a\}_{a \in A}$ over B such that \geq_a specifies the preferences (including ties) of agent a over B , and E is the edge set of a social network between agents in A , i.e., the social network is the undirected graph (A, E) .

We define a matching μ of a given OAF $F = (A, B, \geq, E)$ as a subset of $A \times B$ such that no agent or object belongs to more than one pair in μ . (Put differently, μ is a matching in the complete bipartite graph of agents and objects.) We say that such a matching is perfect if $|\mu| = |A|$. For any matching μ , we define $\text{agents}(\mu)$ (resp., $\text{objects}(\mu)$) as the set of all matched agents (resp., objects) with respect to μ . For any matching μ and any agent a that is matched in μ , we use the shorthand notation $\mu(a)$ to refer to the object matched to agent a . For any matching μ and any object b that is matched in μ , we use the notation $\mu^{-1}(b)$ to refer to the agent matched to object b .

For any OAF $F = (A, B, \geq, E)$, any perfect matching μ of F , and any edge $e = (a, a')$ in E , we define an exchange operation on edge e as $\mu' = \mu \setminus \{(a, \mu(a)), (a', \mu(a'))\} \cup \{(a, \mu(a')), (a', \mu(a))\}$, where μ' is the new matching obtained by applying the exchange operation on edge e . We say the exchange is rational if $\mu(a') \geq_a \mu(a)$ and $\mu(a) \geq_{a'} \mu(a')$, denoted as $\mu \rightarrow_{F,e} \mu'$. We call a rational exchange a *swap*. We use $\mu \rightarrow_F \mu'$ to denote that $\mu \rightarrow_{F,e} \mu'$ for some edge e . We write $\mu \rightsquigarrow_F \mu'$ if there exists a sequence of matchings $\mu = \mu_0, \dots, \mu_k = \mu'$ of matchings of F such that $\mu_{i-1} \rightarrow_F \mu_i$ for $1 \leq i \leq k$.

We define a configuration as a pair $\chi = (F, \mu)$ where F is an OAF and μ is a perfect matching of F . We say that χ is a path or star or tree configuration if the social network in F is a path or star or tree. For any configuration $\chi = (F, \mu)$, we define $\text{reach}(\chi)$ as the set of all perfect matchings μ' of F such that $\mu \rightsquigarrow_F \mu'$.

A matching μ Pareto-dominates a matching μ' if $\mu(a) \geq_a \mu'(a)$ for all agents a in A , and there is at least one agent b in B such that $\mu(b) >_b \mu'(b)$. A matching is Pareto-efficient if it is not Pareto-dominated by another matching. Throughout this paper, we restrict the definition of Pareto-efficiency to the set of reachable matchings. A matching μ is Pareto-efficient with respect to a configuration χ if μ belongs to $\text{reach}(\chi)$ and μ is not Pareto-dominated by another matching in $\text{reach}(\chi)$.

Maximum Votes Pareto-Efficient Matching Problem. Given a configuration $\chi = (F, \mu)$ and a matching μ' , we use $\text{votes}_F(\mu', \mu)$ to denote the number of agents who prefer μ' to μ . We say $\text{votes}_F(\mu', \mu)$ is the voting number of matching μ' . The maximum votes Pareto-efficient matching problem is equivalent to the following optimization problem:

$$\operatorname{argmax}_{\tau \in \text{reach}(\chi) \wedge \tau \text{ is Pareto-efficient}} \text{votes}_F(\tau, \mu).$$

For a given configuration χ , we use $\text{mv}(\chi)$ to denote the largest voting number of a matching in $\text{reach}(\chi)$. We refer to $\text{mv}(\chi)$ as the voting number of χ .

3 Path Network

In this section, we study the maximum votes Pareto-efficient matching problem in a path configuration. We begin by introducing some notations. For any nonnegative integer n , we define $[n]$ as $\{1, \dots, n\}$. Without loss of generality, in this subsection we restrict attention to OAFs of the form $F = ([n], [n], \geq, \{(b, b+1) \mid 1 \leq b < n\})$ for some positive integer n . We use $[i, j]$ to denote the set of agents between agent i and agent j (inclusive). Let $\Pi = \mu_0, \mu_1, \dots, \mu_k$ denote a sequence of matchings where $\mu_i \rightarrow_F \mu_{i+1}$ for $0 \leq i \leq k-1$, i.e., μ_{i+1} is obtained from μ_i by a swap. We have the following observations.

► **Observation 1.** *Each object only moves in one direction or stays at its initial position in Π .*

We say an object is right-moving (left-moving) in Π if it moves to the right (left). The following observation says that while implementing the sequence of swaps Π , the relative location of any two objects moving in the same direction on the path does not change, i.e., if an object b is located to the left (right) of another object b' , then object b will always be to the left (right) of object b' .

► **Observation 2.** *Let $\Pi = \mu_0, \mu_1, \dots, \mu_k$ denote a sequence of matchings. For any two objects b and b' where $\mu_0^{-1}(b) < \mu_0^{-1}(b')$ that move along the same direction in Π , we have $\mu^{-1}(b) < \mu^{-1}(b')$ for all μ in Π .*

Let $\kappa(j, k)$ denote a canonical sequence of exchanges that assigns object j to agent k by directly moving it along the dipath from j to k . Formally, if $j < k$, $\kappa(j, k) = (j, j+1), (j+1, j+2), \dots, (k-1, k)$. If $j \geq k$, $\kappa(j, k) = (j, j-1), (j-1, j-2), \dots, (k+1, k)$. A sequence of exchanges Π is said to be a sequence of swaps if all its exchanges are rational, denoted as $\text{rational}(\Pi)$. We remark that $\kappa(j, k)$ was defined, and some associated properties were proved, by Gourvès et. al [13]. The following lemma presents a useful structural property.

► **Lemma 3.** *Let $\chi = (F, \mu)$ denote a path configuration, let a denote a leaf agent in χ , and let a' denote an agent in χ . If there is a matching μ'' in $\text{reach}(\chi)$ such that $\mu''(a) = \mu(a')$, then (1) $\text{rational}(\kappa(a', a))$ holds, and (2) μ'' belongs to $\text{reach}((F, \mu'))$, where μ' is the matching reached from μ via $\kappa(a', a)$.*

3.1 Maximum Votes Pareto-Efficient Matching

In this section, we present a polynomial-time algorithm for finding a maximum votes Pareto-efficient matching given a path configuration. In order to find an MVPE matching, we first present an algorithm for computing the maximum voting number of a given configuration $\chi = (F, \mu)$, i.e., for computing $\text{mv}(\chi)$. We then compute an MVPE matching by combining

this algorithm with the serial dictatorship algorithm [3]. In the serial dictatorship algorithm, we let agents pick whatever best object they are left with in an arbitrary sequential order. In our case, agents pick objects according to the order of their IDs, i.e., the smallest agent picks first and the largest agent picks last. Assuming that agent a picks object b , we need to meet three additional requirements. First, object b can be assigned to agent a via a sequence of swaps. Second, after object b is assigned to agent a via a sequence of swaps, we obtain a new matching μ' and a new configuration $\chi' = (F, \mu')$. We need to maintain the invariant that there exists a matching τ in $\text{reach}(\chi')$ such that $\text{votes}_F(\tau, \mu) = \text{mv}(\chi)$. Third, object b is the most preferred object of agent a subjects to the first two constraints.

The above three constraints along with the property of serial dictatorship are sufficient for us to prove that the eventual matching we obtain is an MVPE matching. Now, let us first present our algorithm for computing the maximum votes matching of $\text{reach}(\chi)$.

3.2 Maximum Number of Votes

Throughout this section, let $\chi = (F, \mu)$ denote a path configuration. Below we present Algorithm 1, which computes the largest voting number of matching in $\text{reach}(\chi)$.

► **Definition 4** (Directional sequence). *A directional sequence is a sequence of R, L, or S symbols. For any agent i , the i th symbol represents the final moving state of its initial object. Symbol R indicates that it moves to the right. Symbol L indicates that it moves to the left. Symbol S indicates that it does not move.*

For any matching ν in $\text{reach}(\chi)$, we define an associated directional sequence as follows. For agent i , if its initial object $\mu(i)$ moves to the left in ν (i.e., the position $\nu^{-1}(\mu)$ of object μ_i is smaller than the initial position i in μ), then the i th symbol in the directional sequence is L, i.e., the object is left-moving. If it is moved to the right, then the i th symbol is R, i.e., the object is right-moving. If it is stationary, then the i th symbol is S. We use $DS(\nu)$ to denote the directional sequence of ν and $DS(\nu([i, j]))$ to denote the subsequence $DS(\nu)[i, j]$ with respect to agents in $[i, j]$ only.

► **Definition 5** (RL-block). *We define an RL-block as a directional sequence of the form $R^m L^n = RRR\dots RLLL\dots L$ where m, n are positive.*

Remark: Any agent with symbol S in $DS(\nu)$ does not get a better allocation in ν ; any agent with symbol R or L gets improved by ν due to the definition of swaps. Therefore, given a matching ν in $\text{reach}(\chi)$, we can decompose $[n]$ into a set of maximal intervals such that all agents in each interval get moved or improved. Let $\text{Decompose}(\nu)$ denote the set of maximal intervals decomposed from $[n]$ according to ν .

► **Observation 6.** *Given a matching ν in $\text{reach}(\chi)$, $DS(\nu)$ can be uniquely decomposed into a disjoint union of RL-blocks and S symbols.*

Given a matching ν in $\text{reach}(\chi)$, by applying Observation 6 on the directional sequence $DS(\nu)$, we observe that for any interval $[i, j]$ in $\text{Decompose}(\nu)$, the interval $[i, j]$ can be uniquely partitioned as a set of sub-intervals such that for each sub-interval $[i', j']$, $DS(\nu)[i', j']$ is an RL-block. We refer to this partition as the RL-decomposition of interval $[i, j]$.

Given a maximal interval $[i, j]$ in $\text{Decompose}(\nu)$, let $\text{RLD}([i, j])$ denote the set $\{[i', j'] \subseteq [i, j] \mid DS(\nu)[i', j'] \text{ is an RL-block}\}$ that contains those subintervals of $[i, j]$ that are also RL-blocks. Let $\text{RLD}(\nu)$ denote the set $\bigcup_{[i, j] \in \text{Decompose}(\nu)} \text{RLD}([i, j])$.

► **Definition 7** (RL-interval). We say that an interval $[i, j]$ is an RL-interval if at least one of the following condition holds: (1) $\text{rational}(\kappa(j, i))$; (2) $\text{rational}(\kappa(i, j))$; (3) there exists an integer k such that $i < k < j$, $\text{rational}(\kappa(k + 1, i))$, and $\text{rational}(\kappa(k, j))$.

From the above definition, we can obtain the following observation.

► **Observation 8.** For any RL-interval $[i, j]$, there exists a sequence of swaps between agents in $[i, j]$ that improves all agents in the interval.

► **Lemma 9.** Let ν be a matching in $\text{reach}(\chi)$ and let $[i, j]$ be in $\text{RLD}(\nu)$, interval $[i, j]$ is an RL-interval.

Proof. Let $R^m L^{j-i+1-m}$ denote the RL-block $DS(\nu)[i, j]$. Let b_1 and b_2 denote the initial endowment of agent $i + m - 1$ and agent $i + m$ in χ , respectively. Then, b_1 is the rightmost right-moving object and b_2 is the leftmost left-moving object. Then, from Observations 1 and 2, we have $\nu^{-1}(b_1) = j$ and $\nu^{-1}(b_2) = i$; otherwise the initial endowments of agent i and agent j are stationary. Thus, we have a reachable matching ν mapping b_1 to leaf agent j , and mapping b_2 to leaf agent i . By Lemma 3, $\kappa(i + m - 1, j)$ and $\kappa(i + m, i)$ are both rational. ◀

Let \mathcal{I} denote the set of all RL-intervals in χ . Let $\Psi(\mathcal{I})$ denote the set $\{X \subseteq \mathcal{I} \mid \text{all intervals in } X \text{ are disjoint}\}$.

► **Lemma 10.** For any matching ν in $\text{reach}(\chi)$, the set of intervals $\text{RLD}(\nu)$ is an element in $\Psi(\mathcal{I})$.

Proof. Since \mathcal{I} contains all RL-intervals in χ and each interval in $\text{RLD}(\nu)$ is an RL-interval (Lemma 9), the set of intervals $\text{RLD}(\nu)$ is a subset of \mathcal{I} . Also, all intervals in $\text{RLD}(\nu)$ are clearly disjoint. ◀

► **Lemma 11.** For any set of intervals D in $\Psi(\mathcal{I})$, there exists a matching ν in $\text{reach}(\chi)$ such that $\text{RLD}(\nu) = D$.

Proof. We explicitly construct such a matching as follows. For each RL-interval $[i, j]$ in D , from Observation 8, we know there exists a sequence of swaps that improves all agents in $[i, j]$. Thus, matching ν can be constructed by applying such a sequence of swaps for each interval $[i, j]$ in D on the initial matching μ . ◀

Let $\text{MCDI}(\mathcal{I})$ denote the maximum coverage of some disjoint intervals in \mathcal{I} , i.e., the maximum size of the union of disjoint intervals in \mathcal{I} .

► **Lemma 12.** $\text{mv}(\chi) = \text{MCDI}(\mathcal{I})$.

Proof. Lemma 10 implies that $\text{mv}(\chi) \leq \text{MCDI}(\mathcal{I})$. Assume by contradiction that $\text{mv}(\chi) < \text{MCDI}(\mathcal{I})$. Then, there exists a set D in $\Psi(\mathcal{I})$ such that $\text{mv}(\chi) < \sum_{[i,j] \in D} j - i + 1$. By Lemma 11, there exists a matching ν in $\text{reach}(\chi)$ such that $\text{RLD}(\nu) = D$. For matching ν , we have $\text{votes}_F(\nu, \mu) = \sum_{[i,j] \in \text{RLD}(\nu)} j - i + 1 = \sum_{[i,j] \in D} j - i + 1 > \text{mv}(\chi)$, a contradiction. ◀

Using the above lemma, to compute the maximum votes of a configuration, we only need to compute the maximum coverage of disjoint intervals in \mathcal{I} . Algorithm 1 has two steps. The nested for loops compute all the RL-intervals \mathcal{I} for a given χ . The second step invokes the maximum disjoint interval algorithm on \mathcal{I} .

Algorithm 1 $MV(\chi)$.

Input: A configuration $\chi = (F, \mu_0)$
Output: The maximum votes number of χ

```

 $\mathcal{I} \leftarrow \emptyset$ 
for  $i \leftarrow 1$  to  $n - 1$  do
  for  $j \leftarrow i + 1$  to  $n$  do
    if  $\exists k \in [i, j]$  s.t.  $\text{rational}(\kappa(k, j)) \wedge \text{rational}(\kappa(k + 1, i))$  then
       $\mathcal{I} \leftarrow \mathcal{I} \cup \{[i, j]\}$ 
    end
  end
end
Return  $MCDI(\mathcal{I})$ 

```

3.3 Weighted Version of Maximum Votes Matching

We remark that there is a polynomial-time algorithm solving a weighted version of finding a maximum votes matching. Formally, let $W : [n] \rightarrow \mathbb{R}$ denote a weight function assigning each agent a in $[n]$ a real value $W(a)$. For any matchings μ', μ of F , let $\text{votes}_F(\mu', \mu, W) = \sum_{a \in [n]; \mu'(a) >_a \mu(a)} W(a)$, i.e., the total weight of the agents that prefer μ' to μ . In addition, let $\text{mv}(\chi, W) = \max_{\mu' \in \text{reach}(\chi)} \text{votes}_W(\mu', \mu)$.

To allow for the weight function W , we reduce the problem of computing $\text{mv}(\chi, W)$ to compute the maximal sum of weights of disjoint intervals in \mathcal{I} , where each interval $[i, j]$ in \mathcal{I} has weight $\sum_{k \in [i, j]} W(k)$. To compute the maximum weighted coverage of disjoint intervals in \mathcal{I} , we construct a directed graph $G = (\mathcal{I}, E)$ as follows. Each node in G is a interval $[i, j]$ in \mathcal{I} . For every two nodes $[i, j]$ and $[i', j']$, there is a directed edge from node $[i, j]$ to node $[i', j']$ if $j < i'$. Moreover, each node has a cost equal to the weight of the corresponding interval. Note that this graph is acyclic, and it takes $O(n^2)$ time to find a most weighted path in an acyclic graph.

3.4 MVPE algorithm

In this section, we present Algorithm 3 for finding an MVPE matching given a path configuration. Our main idea is to apply a serial dictatorship procedure. To simplify the presentation, we focus on the leftmost agent a in any path configuration $\chi = (F, \mu)$ on agents $[a, n]$. We seek the best possible object b for the leftmost agent a such that there exists a maximum votes matching τ in $\text{reach}(\chi)$ with $\tau(a) = b$. Note that Lemma 3 implies that $\text{rational}(\kappa(\mu^{-1}(b), a))$ holds. We then apply $\kappa(\mu^{-1}(b), a)$ in χ to let agent a match object b . Hence, we get a new configuration χ' on agents $[a + 1, n]$ by truncating the leftmost agent a along with its assigned object b . We then recurse on the leftmost agent $a + 1$ in χ' .

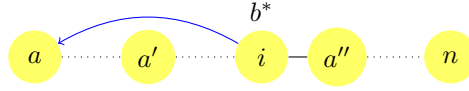
However, when we recurse on χ' , the situation is a bit different than in χ . The applied sequence of swaps $\kappa(\mu^{-1}(b), a)$ has improved agents between agent a and $\mu^{-1}(b)$ in χ , and also in χ' . These agents will vote when this serial dictatorship procedure ends. Thus, when we recurse on χ' , in order to ensure that the resulting matching is a maximum vote matching, we need to find a matching maximizing votes among agents on the right side of $\mu^{-1}(b)$ in χ' , rather than all agents.

Formally, we introduce some notations. For any agent interval $[i, j]$ that is a subset of $[n]$, let $\chi[i, j]$ and $\mu[i, j]$ denote the truncated configuration and truncated matching induced by agents in $[i, j]$, respectively. Let $\text{mv}(\chi, [i, j])$ denote the maximum number of agents in $[i, j]$

that can be improved by any reachable matching in $\text{reach}(\chi)$, $\text{votes}_F(\mu', \mu, [i, j])$ denote the number of agents in $[i, j]$ that prefer μ' to μ , and $\text{MVM}(\chi, [i, j])$ denote the set of reachable matchings achieving $\text{mv}(\chi, [i, j])$.

We are prepared to introduce our recursive subroutine called BOLA (best object for the leftmost agent), shown as Algorithm 2. Given a configuration $\chi = (F, \mu)$ on agents $[a, n]$ and an agent $a' \geq a$, Algorithm 2 finds the leftmost agent a 's most preferred object b^* such that there exists a μ' in $\text{MVM}(\chi, [a', n])$ with $\mu'(a) = b^*$. In Algorithm 2, agent a' is used to limit the objects that agent a can be possibly matched with. Specifically, Algorithm 2 only considers objects that are initially owned by agents in $[a', n]$. In Algorithm 2, we need to compute $\text{mv}(\chi, [a, n])$ for a given χ on the agents in $[a, n]$. This can be done by applying the weighted version of Algorithm 1 as discussed in Section 3.3.

Fig. 4 shows a possible execution of Algorithm 2. Here, Algorithm 2 runs with a configuration χ on the agents in $[a, n]$ and an agent a' in $[a, n]$, and it finds an object b^* initially owned by agent i that satisfies the two **if** conditions.



■ **Figure 4** A demo example for Algorithm 2.

■ **Algorithm 2** $\text{BOLA}(\chi, a')$.

Input: A configuration $\chi = (F, \mu)$ with agents $[a, n]$, an agent $a' \in [a, n]$
Output: The most preferred object b^* of the leftmost agent, where there is a matching μ' in $\text{MVM}(\chi, [a', n])$ assigning b^* to a

// Remark: a is the leftmost agent in χ
 $b^*, V \leftarrow \mu(a), \text{mv}(\chi, [a', n])$
for $i \leftarrow a$ **to** n **do**
 if $\mu(i) \geq_a b^* \wedge \text{rational}(\kappa(i, a))$ **then**
 // Remark: i denotes an agent
 $\mu^* \leftarrow$ the matching that results from applying $\kappa(i, a)$ to μ
 $\chi^* \leftarrow (F, \mu^*)$
 $a'' \leftarrow \max(i + 1, a')$
 if $\text{votes}(\mu^*, \mu, [a', a'' - 1]) + \text{mv}(\chi^*, [a'', n]) = V$ **then**
 $b^* \leftarrow \mu(i)$
 end
 end
end
return b^*

Lemma 13 characterizes all possible maximum votes matchings in terms of the two **if** conditions used in Algorithm 2. Next we use Lemma 13 to prove Lemma 14, which is our main correctness lemma. It implies that to find the leftmost agent a 's most preferred object b^* such that $\exists \mu' \in \text{MVM}(\chi, [a', n]) : \mu'(a) = b^*$, we can iterate through all objects that satisfy the two **if** conditions and find the best one for agent a .

► **Lemma 13.** *Let $\chi = (F, \mu)$ denote an configuration, let $[a, n]$ denote the set of agents in χ , and let a' denote an agent in $[a, n]$. Let τ denote a matching in $\text{reach}(\chi)$. Let $b = \tau(a)$, and $a'' = \max(\mu^{-1}(b) + 1, a')$. Then, the matching τ belongs to $\text{MVM}(\chi, [a', n])$ if and only if the following two conditions hold:*

- (1) $\text{rational}(\kappa(\mu^{-1}(b), a))$;
- (2) $\text{votes}(\mu', \mu, [a', a'' - 1]) + \text{mv}(\chi'[a + 1, n], [a'', n]) = \text{mv}(\chi, [a', n])$, where μ' is obtained from applying $\kappa(\mu^{-1}(b), a)$ to μ and $\chi' = (F, \mu')$.

► **Lemma 14.** *Algorithm 2 with an input χ on agents $[a, n]$ and an agent a' in $[a, n]$ returns the most preferred object b^* of agent a 's such that there exists a matching μ' in $\text{MVM}(\chi, [a', n])$ with $\mu'(a) = b^*$.*

Proof. It is straightforward to verify that the returned b^* satisfies two conditions (1) and (2) in Lemma 13. Therefore, Lemma 13 implies that there exists a matching μ' in $\text{MVM}(\chi, [a', n])$ with $\mu'(a) = b^*$.

Then we prove that b^* is the agent a 's most preferred object. Assume by contradiction that there is another b is more most preferred to agent a than b^* and $\exists \mu' \in \text{MVM}(\chi, [a', n]) : \mu'(a) = b$. By Lemma 13, two conditions (1) and (2) in Lemma 13 holds and thus the algorithm will return b instead, a contradiction. ◀

We now present Algorithm 3 that uses Algorithm 2 as a building block to find an MVPE matching. In Algorithm 3, we maintain a partial matching τ_i , which is a matching involving only agents in $[1, i]$. At the i th iteration, we invoke the BOLA subroutine to find the agent i 's most preferred object b^* that satisfies the condition in Lemma 14. Then we apply $\kappa(b^*, i)$ to assign object b^* to agent i and remove agent i from the configuration afterwards.

Let $P(i)$ denote the predicate: there exists ν in $\text{MVM}(\chi, [1, n])$ such that τ_i is a subset of ν . Algorithm 3 maintains the invariant: $P(i)$ holds for all i in $[n]$. This invariant ensures that the final matching τ_n returned by Algorithm 3 is a matching in $\text{MVM}(\chi, [1, n])$. The serial dictatorship mechanism ensures that τ_n is also a Pareto-efficient matching.

■ **Algorithm 3** $\text{MVPEM}(\chi)$.

Input: A configuration $\chi = (F, \mu)$
Output: An MVPE matching of χ

$\mu_1, a_1, \chi_1, \tau_0 \leftarrow \mu, 1, \chi, \emptyset$
for $i \leftarrow 1$ **to** n **do**
 /* i is the index of an agent */
 $b_i \leftarrow \text{BOLA}(\chi_i, a_i)$
 $a'_i \leftarrow \mu_i^{-1}(b_i)$
 $a_{i+1} \leftarrow \max(a'_i + 1, a_i)$
 $\mu' \leftarrow$ the matching that results from applying $\kappa(a'_i, i)$ to μ_i
 /* Assign b_i to agent i and truncate it from χ */
 $\tau_i, \mu_{i+1}, \chi_{i+1} \leftarrow \tau_{i-1} + (i, b_i), \mu'[i + 1, n], (F, \mu_{i+1})$
end
return τ_n

► **Theorem 15.** *The matching τ_n returned by Algorithm 3 is an MVPE matching.*

4 Star Network

In this section, we consider the case when the social network is a star. Our results are twofold. We first present a polynomial-time algorithm for finding an MVPE matching in a star network when preferences are strict. We then show that finding an MVPE matching in a star network with weak preferences is NP-hard.

For the case of strict preferences, we present a simple polynomial-time algorithm for finding an MVPE matching. The main idea of the algorithm is to use the fact that any swap involves a center agent and a non-center agent to reduce the MVPE problem to the problem of finding a longest path in a directed acyclic graph.

► **Theorem 16.** *There is a polynomial-time algorithm finding an MVPE matching in a star network with strict preferences.*

Proof. Let O denote the center agent, and let $[n]$ denote the set of leaf agents. Let μ denote the initial matching. Note that every swap in a star network always involves the center agent. Moreover, notice that any leaf agent can change its object at most once in any valid sequence of swaps. To see this, observe that once an object moves from the center to a leaf, it can never return to the center. Thus, any sequence of swaps can be represented as an ordered list of leaf agents. Let $L = (i_1, i_2, \dots, i_k)$ denote such a list, where k belongs to $[n]$. Note that $\mu(i_j) <_O \mu(i_{j+1})$ for all j in $[k-1]$, where O is the center agent.

We first show how to find a maximum votes matching, and then prove that any maximum votes matching is also Pareto-efficient.

To find a maximum votes matching in star networks, it is equivalent to find a longest ordered list of leaves corresponding to a sequence of swaps, since a leaf agent gets improved if and only if it is included in the list. This problem reduces to the search of a longest path in a directed acyclic graph $G = (\{O\} \cup [n], E')$. The edge set E' contains a directed edge from a to a' if and only if agent a' belongs to $[n]$, $\mu(a') >_O \mu(a)$, and $\mu(a) >_{a'} \mu(a')$. There is a path from O to a' in the digraph if and only if the center can get the initial endowment of agent a' . It is straightforward to see that there exists a polynomial-time algorithm solving this longest path problem in G .

Next we prove that any maximum votes matching ν is Pareto-efficient. Assume for the sake of contradiction that reachable matching ν' Pareto-dominates ν . Consider the ordered lists of agents L and L' corresponding to the sequences of swaps for reaching ν and ν' , respectively. Since ν' Pareto-dominates ν , any agent in L that gets a better allocation in ν also gets a better allocation in ν' , i.e., belongs to L' . Furthermore, since ν is a maximum votes matching, the ordered list L is the longest ordered list of agents. Therefore, the list L' is contained in L . (Otherwise, L' contains all agents in L and is longer than L , which contradicts the assumption that L is the longest ordered list of agents.) That is, L' has the same agents as in L , and hence $L' = L$ as all agents in the ordered lists are sorted by the preferences of center agent O . Therefore, we have $\nu' = \nu$, a contradiction. ◀

4.1 Weak Preference

IN this section, we show that finding an MVPE matching is NP-hard in a star network with weak preferences. We first introduce the notion of a center object sequence. For all sequences of matchings $\Pi = \mu_0, \dots, \mu_k$, we define the corresponding center object sequence π as the list of objects owned by the center agent O , i.e., $\pi = [\mu_0(O), \dots, \mu_k(O)]$. We use $\pi[i]$ to denote $\mu_i(O)$, and for any object b , we use $\pi(b)$ to denote the index of first occurrence of b in π , and $\pi(b) = -1$ if b does not belong to π . We have the following observations for the center object sequence.

► **Observation 17.** *Let X be a non-center agent with initial endowment x such that x belongs to π . Then, $\pi[\pi(x) - 1] \geq_X x$.*

Notice that $\pi[\pi(x) - 1]$ is the object used by center agent O to swap x from agent X .

► **Observation 18.** *Let X be an agent with initial object x . Then, agent X is improved by Π if and only if x belongs to π .*

We now establish NP-hardness by proving that it is NP-hard to find a maximum votes matching.

Construction. We use a reduction from the 3-SAT. In an instance of 3-SAT, we are given a propositional formula ϕ that is the conjunction of m clauses C_1, \dots, C_m . Each clause C_i is the disjunction of three literals, where each literal is either a variable or the negation of a variable. The set of variables is x_1, \dots, x_n . Given a formula ϕ , we construct a corresponding star configuration $\chi_\phi = (F, \mu)$ with $3n + 3m + 1$ agents such that for any 3-SAT formula ϕ with n variables and m clauses, ϕ is satisfiable if and only if χ_ϕ has largest voting number $2n + 3m$.

We begin by creating the set of agents A . For each variable index i , there are three agents in A : S_i, T_i, F_i . For each clause $C_j = l_{j,0} \cup l_{j,1} \cup l_{j,2}$, there are three agents $P_{j,0}, P_{j,1}, P_{j,2}$ in A . There is also a center agent O . Thus there are a total of $3m + 3n + 1$ agents in A .

For each agent in A , we use the corresponding lower case letter to denote its initial object. For example, agent $P_{1,2}$ initially holds object $p_{1,2}$. For agents T_i and F_i in a variable gadget with $i \in [n]$, their initial objects are t_i and f_i , representing whether the corresponding variable x_i is true or false. Furthermore, for each literal $l_{j,k}$ of variable x_w , we associate instance an object $a_{j,k}$ defined as follows. If $l_{j,k} = x_w$, then $a_{j,k} = t_w$; otherwise, $a_{j,k} = f_w$. For example, if $l_{j,k} = x_{15}$, then $a_{j,k} = t_{15}$, and if $l_{j,k} = \bar{x}_7$, then $a_{j,k} = f_7$.

For agent preferences, we only consider the objects that each agent prefers at least as its initial object; other objects can be put behind its initial endowment in any order. The center agent O is indifferent to all objects. We use boxed objects to indicate the initial endowments, and objects in a bracket are indifferent to an agent. For agents S_i, T_i, F_i , their preferences from most preferred to least preferred are $S_i : (t_i, f_i), (o, \boxed{s_i})$, $T_i : s_i, \boxed{t_i}$, $F_i : s_i, \boxed{f_i}$. For each k in $\{0, 1, 2\}$, $P_{j,k} : p_{j,k-1}, a_{j,k}, \boxed{p_{j,k}}$, where $p_{j,-1} = p_{j,2}$.

We now present some properties of our gadgets.

► **Lemma 19.** *Let S_i, T_i, F_i be the three agents in the variable gadget corresponding to a variable x_i , $\Pi = \mu_0, \dots, \mu_k$ be a sequence of matchings, and π denote the corresponding center object sequence of Π . Then at most two agents in $\{S_i, T_i, F_i\}$ are improved by Π . If exactly two agents of $\{S_i, T_i, F_i\}$ are improved, then exactly one of $\{t_i, f_i\}$ belongs to π .*

Proof. Notice that if agent T_i or F_i is improved, then either T_i or F_i gets object s_i by the preferences of T_i and F_i . However, in a matching, s_i can be matched to one agent. Therefore, Π cannot improve T_i and F_i . That is, at most two agents in $\{S_i, T_i, F_i\}$ are improved by Π . If exactly two agents of $\{S_i, T_i, F_i\}$ get improved, then exactly one agent in $\{T_i, F_i\}$ is improved, and hence exactly one of $\{t_i, f_i\}$ belongs to π by Observation 18. ◀

► **Lemma 20.** *Let $C_j = l_{j,0} \cup l_{j,1} \cup l_{j,2}$ denote a clause. For each k in $\{0, 1, 2\}$, let $P_{j,k}$ denote the agent corresponding to literal $l_{j,k}$, let $p_{j,k}$ denote the initial object of $P_{j,k}$, and let $a_{j,k}$ in $\{t_i, f_i \mid i \in [n]\}$ denote the object associated with literal $l_{j,k}$. Let $\Pi = \mu_0, \dots, \mu_k$ be a sequence of matchings, and π denote the corresponding center object sequence of Π . If none of $a_{j,0}, a_{j,1}, a_{j,2}$ is in π , then none of $p_{j,0}, p_{j,1}, p_{j,2}$ is in π .*

Proof. We prove the contrapositive. Assume that there is an object in $\{p_{j,0}, p_{j,1}, p_{j,2}\}$ that belongs to π . Let $p_{j,k}$ denote the object with minimum $\pi(p_{j,k})$ where k belongs to $\{0, 1, 2\}$. By Observation 17, $\pi[\pi(p_{j,k}) - 1] \geq_{P_{j,k}} p_{j,k}$. By the preferences of $P_{j,k}$, $\pi[\pi(p_{j,k}) - 1]$ belongs

to $\{a_{j,k}, p_{j,k-1}\}$ with $p_{j,-1} = p_{j,2}$. Moreover, by the assumption $p_{j,k}$ is the object with minimum $\pi(p_{j,k})$, $\pi[\pi(p_{j,k}) - 1]$ does not belong to $\{p_{j,0}, p_{j,1}, p_{j,2}\}$. Therefore, $\pi[\pi(p_{j,k}) - 1]$ is $a_{j,k}$, and hence π contains object $a_{j,k}$. \blacktriangleleft

Using the above properties of our gadgets, it is not hard to verify the correctness of our reduction. Formally, we prove following two lemmas, one for each reduction direction.

► **Lemma 21.** *If $\text{mv}(\chi_\phi) = 2n + 3m$, then ϕ is satisfiable.*

Proof. By Lemma 19, there are at least n agents that are not improved. Moreover, the center agent O initially holds one of most preferred objects, and hence cannot be improved. Therefore, there are at most $2n + 3m$ agents that are improved. If χ_ϕ achieves the largest voting number $2n + 3m$, then there are exactly two agents that are improved in each variable gadget and all three agents that are improved in each clause gadget.

By Lemma 19, for all i in $[n]$, there is exactly one object in $\{t_i, f_i\}$ in π . If t_i is in π , then let variable x_i be true, otherwise let x_i be false. Therefore, we get an assignment A_π from π . We then show that A_π is a satisfiable assignment, i.e., each clause C_j in ϕ is true with respect to assignment A_π . For the sake of contradiction, assume that there exists a clause C_j that is false. Hence, all of the literals $l_{j,0}, l_{j,1}, l_{j,2}$ in C_j are false under assignment A_π . Let $a_{j,0}, a_{j,1}, a_{j,2} \in \{t_i, f_i \mid i \in [n]\}$ be the objects associated with literals $l_{j,0}, l_{j,1}, l_{j,2}$, respectively. We deduce that none of $a_{j,0}, a_{j,1}, a_{j,2}$ is in π . By Lemma 20, π does not contain $\{p_{j,0}, p_{j,1}, p_{j,2}\}$. That is, agents $P_{j,0}, P_{j,1}, P_{j,2}$ is not improved by π due to Observation 18. \blacktriangleleft

► **Lemma 22.** *If there is a satisfiable assignment for ϕ , then $\text{mv}(\chi_\phi) = 2n + 3m$.*

Proof. Let A_ϕ be a satisfiable assignment for ϕ . For each i in $[n]$, if x_i is true in assignment A_ϕ , then let b_i denote object t_i ; otherwise, let b_i denote object f_i .

Now we show how to obtain a sequence of swaps according to objects (b_1, \dots, b_n) . Let \mathcal{C} denote a temporary set, which is initialized to be the set of all clauses $\{C_1, C_2, \dots, C_m\}$. For each i in $[n]$, let B_i denote the initial owner of b_i , i.e., $B_i = T_i$ if $b_i = t_i$ and $B_i = F_i$ otherwise. We construct the sequence of swaps as follows. For each object b_i , we do the following swaps:

1. Perform two swaps $(O, S_i), (O, B_i)$ to let center agent O get b_i and S_i get object o .
2. If there are a clause C_j in \mathcal{C} and a literal $l_{j,k}$ in C_j such that object b_i is the associated object $a_{j,k}$, then we perform the following steps:
 - Perform these swaps in sequence: $(O, P_{j,k}), (O, P_{j,k \oplus 1}), (O, P_{j,k \oplus 2}), (O, P_{j,k})$, where \oplus denotes the addition module 3. Remark that by doing so, agent $P_{j,k}$ gets its most preferred object for all k in $\{0, 1, 2\}$ and center agent O still holds b_i .
 - Remove the clause C_j from \mathcal{C} .
3. In the end, perform the swap (O, S_i) . Remark: After this swap, agent O gets object o and S_i get one of its most preferred object b_i . We then proceed to consider object b_{i+1} .

We now show that the sequence of swaps constructed above improves $2n + 3m$ agents. First of all, by applying the above sequence of swaps, for all i in $[n]$, agent S_i is matched to object b_i and agent B_i is matched to object s_i . Thus, the above sequence of swaps improves $2n$ agents in variable gadgets. Then, we consider agents in clause gadgets. Since A_ϕ is a satisfiable assignment for ϕ , for all clause C_j there exists some literal $l_{j,k}$ of variable x_i such that $l_{j,k}$ is true under assignment A_ϕ . Therefore, by the definition of object $a_{j,k}$, we deduce that $a_{j,k}$ is b_i . Thus, all agents associated with clause C_j are improved. That is, the above sequence of swaps improves all agents in clause gadgets. There are $3m$ agents in clause gadgets. To sum up, there are $2n + 3m$ agents that are improved in total. \blacktriangleleft

► **Theorem 23.** *Finding an MVPE matching on stars with weak preferences is NP-hard.*

5 Tree Network

► **Theorem 24.** *Finding an MVPE matching on trees with strict preferences is NP-hard.*

We establish Theorem 24 by proving that it is NP-hard to find a maximum votes matching. The latter result is obtained via a reduction to reachable object problem on trees, which was shown to be NP-complete by Gourvès et al. [13].

6 Conclusion



To refine Pareto-efficiency, we study the complexity of finding a maximum votes Pareto-efficient matching when a group of agents exchange their objects along a social network. By presenting two polynomial-time algorithms and two NP-hardness results, we shed light on the frontiers between tractable and intractable cases in terms of the structures of the social network and whether ties in preference are allowed. A challenging open question is to completely characterize the social network structures for which case the MVPE problem is polynomial-time solvable.

References

- 1 Atila Abdulkadiroğlu, Yeon-Koo Che, and Yosuke Yasuda. Resolving conflicting preferences in school choice: The “Boston Mechanism” reconsidered. *American Economic Review*, 101(1):399–410, 2011.
- 2 Atila Abdulkadiroğlu and Tayfun Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.
- 3 Atila Abdulkadiroğlu and Tayfun Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233–260, 1999.
- 4 David J. Abraham, Katarína Cechlárová, David F. Manlove, and Kurt Mehlhorn. Pareto optimality in house allocation problems. In *Proceedings of the 15th International Symposium on Algorithms and Computation*, pages 3–15, 2004.
- 5 David J. Abraham, Robert W. Irving, Telikepalli Kavitha, and Kurt Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.
- 6 Haris Aziz, Serge Gaspers, Simon Mackenzie, and Toby Walsh. Fair assignment of indivisible objects under ordinal preferences. *Artificial Intelligence*, 227:71–92, 2015.
- 7 Matthias Bentert, Jiehua Chen, Vincent Froese, and Gerhard J. Woeginger. Good things come to those who swap objects on paths. *arXiv*, 2019. [arXiv:1905.04219](https://arxiv.org/abs/1905.04219).
- 8 Aurélie Beynier, Yann Chevaleyre, Laurent Gourvès, Ararat Harutyunyan, Julien Lesca, Nicolas Maudet, and Anaëlle Wilczynski. Local envy-freeness in house allocation problems. *Autonomous Agents and Multi-Agent Systems*, 33(5):591–627, 2019.
- 9 Vittorio Bilò, Ioannis Caragiannis, Michele Flammini, Ayumi Igarashi, Gianpiero Monaco, Dominik Peters, Cosimo Vinci, and William S. Zwicker. Almost envy-free allocations with connected bundles. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference*, pages 14:1–14:21, 2018.
- 10 Péter Biró and Jens Gudmundsson. Complexity of finding Pareto-efficient allocations of highest welfare. *European Journal of Operational Research*, 291(2):614–628, 2021.
- 11 Ágnes Cseh. Trends in computational social choice. In U. Endriss, editor, *Trends in Computational Social Choice*, chapter 6, pages 105–122. lulu.com, 2017.
- 12 Federico Echenique, Antonio Miralles, and Jun Zhang. Fairness and efficiency for probabilistic allocations with endowments. *arXiv*, 2019. [arXiv:1908.04336](https://arxiv.org/abs/1908.04336).
- 13 Laurent Gourvès, Julien Lesca, and Anaëlle Wilczynski. Object allocation via swaps along a social network. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 213–219, 2017.

- 14 Sen Huang and Mingyu Xiao. Object reachability via swaps along a line. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 2037–2044, 2019.
- 15 Sen Huang and Mingyu Xiao. Object reachability via swaps under strict and weak preferences. *Autonomous Agents and Multiagent Systems*, 34(2):51, 2020.
- 16 Ayumi Igarashi and Dominik Peters. Pareto-optimal allocation of indivisible goods with connectivity constraints. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 2045–2052, 2019.
- 17 Telikepalli Kavitha, Julián Mestre, and Meghana Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412(24):2679–2690, 2011.
- 18 Telikepalli Kavitha, Meghana Nasre, and Prajakta Nimbhorkar. Popularity at minimum cost. *Journal of Combinatorial Optimization*, 27(3):574–596, 2014.
- 19 Richard Matthew McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of 8th Latin American Symposium on Theoretical Informatics*, pages 593–604, 2008.
- 20 Luis Müller and Matthias Bentert. On reachable assignments in cycles and cliques. *arXiv*, 2020. [arXiv:2005.02218](https://arxiv.org/abs/2005.02218).
- 21 Abdallah Saffidine and Anaëlle Wilczynski. Constrained swap dynamics over a social network in distributed resource reallocation. In *Proceedings of the 11th International Symposium on Algorithmic Game Theory*, pages 213–225, 2018.
- 22 Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.
- 23 Ryo Yoshinaka. Higher-order matching in the linear lambda calculus in the absence of constants is NP-complete. In Jürgen Giesl, editor, *Proceedings of 16th International Conference on Term Rewriting and Applications*, pages 235–249, 2005.

Finite Models for a Spatial Logic with Discrete and Topological Path Operators

Sven Linker  

Lancaster University in Leipzig, Germany

Fabio Papacchini  

University of Liverpool, UK

Michele Sevegnani  

University of Glasgow, UK

Abstract

This paper analyses models of a spatial logic with path operators based on the class of neighbourhood spaces, also called pretopological or closure spaces, a generalisation of topological spaces. For this purpose, we distinguish two dimensions: the type of spaces on which models are built, and the type of allowed paths. For the spaces, we investigate general neighbourhood spaces and the subclass of quasi-discrete spaces, which closely resemble graphs. For the paths, we analyse the cases of quasi-discrete paths, which consist of an enumeration of points, and topological paths, based on the unit interval. We show that the logic admits finite models over quasi-discrete spaces, both with quasi-discrete and topological paths. Finally, we prove that for general neighbourhood spaces, the logic does not have the finite model property, either for quasi-discrete or topological paths.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Mathematics of computing → Topology

Keywords and phrases spatial logic, topology, finite models

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.72

Funding *Fabio Papacchini*: partly supported by the EPSRC through grants EP/R026084 and EP/R026173.

Michele Sevegnani: supported by the EPSRC under PETRAS SRF grant MAGIC (EP/S035362/1).

1 Introduction

The safe and correct operation of systems in a wide range of application domains is increasingly dependent on spatial reasoning to evaluate the structure of space and how space might evolve over time. Examples include target counting in wireless sensor networks [19, 2], cyber-physical systems [22], transport systems [9], structural analysis [17], and medical imaging [6]. *Neighbourhood spaces*, also known as closure or pretopological spaces [23, 14], have emerged as a popular formalism in these scenarios due to their ability to natively represent topological spaces but also simple graphs and simple directed graphs. In this paper, we focus on *SLCS*, a modal logic introduced by Ciancia et al. [11] for the specification and verification of spatial properties over neighbourhood spaces. This logic features a closure modality \mathcal{N} (near) and path modalities \mathcal{R} (reachable from) and \mathcal{P} (propagates to). While model checking algorithms and software support have been developed, the model theory of this logic is still not well understood. In particular, it is not known what kind of spaces can be expressed by various classes of formulas. Answering this question is complicated by how the near modality interacts with the path modalities which is substantially different from the modality interactions in discrete modal logic.



© Sven Linker, Fabio Papacchini, and Michele Sevegnani;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 72; pp. 72:1–72:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We make the following research contributions:

1. we show that SLCS does not admit finite models on general neighbourhood spaces;
2. we prove that there are formulas that are only satisfiable on infinite models even when restricting to either quasi-discrete paths (similar to paths on graphs) or standard topological paths;
3. we define a finite model construction using filtration arguments for models with quasi-discrete underlying spaces and quasi-discrete or topological paths.

Related Work

The analysis of SLCS is increasingly gaining traction both in Theoretical Computer Science and Topology.

In recent work [18], we presented bisimulations for SLCS formulas using path operators that show the equivalence of formulas between bisimilar models. Ciancia et al. [12] used co-algebraic methods to present bisimulations over quasi-discrete models that are well-matched (i.e., they characterise the class of quasi-discrete models), but did not extend this result to arbitrary spaces. Importantly, the authors restricted the set of SLCS formulas to omit path operators. Castelnovo and Miculan [7] defined a categorical semantics for various fragments of SLCS using hyperdoctrines with paths and investigated how to extend the logic to other spaces with closure operators, such as probabilistic automata.

Rieser [20] used the unit interval to define and analyse a homotopy theory for closure spaces, that is, how paths can be transformed into one another. Bubenik and Milićević [5] further investigated how different generalisations of the unit interval yield different path objects. None of these definitions is immediately applicable to SLCS paths, which are much more concrete.

2 Neighbourhood Spaces

In this section we recall the notions of neighbourhood spaces and some related results from general topology we will use in this paper. Our main reference is [23]. For additional general results on these topics and for the proofs of the results reported here, we refer the reader to this source.

► **Definition 1 (Filter).** *Given a set X , a filter F on X is a subset of $\mathbb{P}(X)$, such that F is closed under intersections, whenever $Y \in F$ and $Y \subseteq Z$, then also $Z \in F$, and finally $\emptyset \notin F$.*

► **Definition 2 (Neighbourhood Space).** *Let X be a set, and let $\eta: X \rightarrow \mathbb{P}(\mathbb{P}(X))$ be a function from X to the set of filters on it, where every $\eta(x)$ is such that $x \in \bigcap_{N \in \eta(x)} N$. We call η a neighbourhood system on X , and $\mathcal{X} = (X, \eta)$ a neighbourhood space. For every set $A \subseteq X$, we have the (unique) interior and closure operators defined as follows.*

$$\mathcal{I}_\eta(A) = \{x \in A \mid A \in \eta(x)\} \quad \mathcal{C}_\eta(A) = \{x \in X \mid \forall N \in \eta(x): A \cap N \neq \emptyset\}$$

An element $x \in X$ has a minimal neighbourhood if there exists $N \in \eta(x)$ such that $N \subseteq N'$ for any neighbourhood $N' \in \eta(x)$. We use $N_{\min}(x)$ to refer to the minimal neighbourhood of x . If each element $x \in X$ has a minimal neighbourhood, then we call \mathcal{X} quasi-discrete. Finally, if for every element $x \in X$ and any neighbourhood $N \in \eta(x)$, there is a neighbourhood $M \in \eta(x)$, such that for every $y \in M$, we have also that $N \in \eta(y)$, then \mathcal{X} is topological.

Neighbourhood spaces as we introduced them are exactly the *pretopological spaces* as defined by Choquet [8] and the *closure spaces* introduced by Čech [23], as shown by Kent and Min [16].¹ Furthermore, a topological neighbourhood space is just a topological space as usual.

► **Definition 3** (Connectedness ([23] 20.B.1)). *Let $\mathcal{X} = (X, \eta)$ be a neighbourhood space. Two subsets U and V of X are semi-separated, if $\mathcal{C}(U) \cap V = U \cap \mathcal{C}(V) = \emptyset$. A subset U of \mathcal{X} is connected, if it is not the union of two non-empty, semi-separated sets. The space \mathcal{X} is connected, if X is connected.*

We also introduce a special kind of connected neighbourhood space, endowed with a linear order.

► **Definition 4** (Index Space). *If (I, η) is a connected neighbourhood space and $\leq \subseteq I \times I$ a linear order on I with the bottom element $0 \in I$, then we call $\mathcal{I} = (I, \eta, \leq, 0)$ an index space.*

In the following sections, we will often use the concept of continuous functions. Generally, we will use the notation $f[A]$ for the image of a set $A \subseteq X$ under a function $f: X \rightarrow Y$.

► **Definition 5** (Continuous Function ([23] 16 A.4)). *Let $\mathcal{X}_i = (X_i, \eta_i)$ for $i \in \{1, 2\}$ be two neighbourhood spaces. A function $f: X_1 \rightarrow X_2$ is continuous at x_1 , if for every $N_2 \in \eta_2(f(x_1))$, there is an $N_1 \in \eta_1(x_1)$ such that $f[N_1] \subseteq N_2$. Equivalently, for every $Y \subseteq X_1$, if $x_1 \in \mathcal{C}_1(Y)$, then $f(x_1) \in \mathcal{C}_2(f[Y])$. If f is continuous at every $x_1 \in X_1$, we simply say that f is continuous. We will also write $f: \mathcal{X}_1 \rightarrow \mathcal{X}_2$.*

Observe that this coincides with the well-known definition of continuous functions on topological spaces.

► **Definition 6** (Path). *For an index space \mathcal{I} and a neighbourhood space \mathcal{X} , a continuous function $p: \mathcal{I} \rightarrow \mathcal{X}$ is an \mathcal{I} -path on \mathcal{X} . If $p(0) = x$, we will also write $p: x \rightsquigarrow \infty$ to denote a path starting in x .*

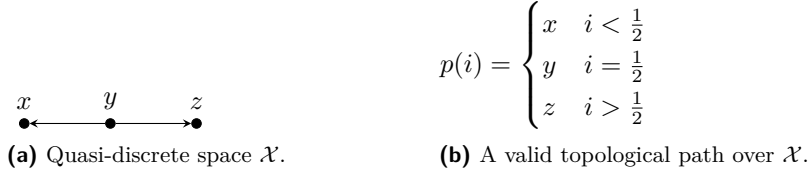
Two typical index spaces are $\mathcal{I}_{\mathbb{R}} = ([0, 1], \eta_{\mathbb{R}}, \leq, 0)$, the unit interval with the standard topology based on open intervals, and $\mathcal{I}_{\mathbb{N}} = (\mathbb{N}, \eta_{\mathbb{N}}, \leq, 0)$, where $\eta_{\mathbb{N}}$ is given by the quasi-discrete neighbourhood system induced by the successor relation. That is, the minimal neighbourhood of each point n is given by $\{n, n + 1\}$. We call $\mathcal{I}_{\mathbb{R}}$ -paths *topological paths* and $\mathcal{I}_{\mathbb{N}}$ -paths *quasi-discrete paths*.

► **Definition 7** (Separation and Distinguishability). *Let $\mathcal{X} = (X, \eta)$ be a neighbourhood space and $x, y \in X$ be two distinct points of \mathcal{X} . If $\eta(x) \neq \eta(y)$, we say that x and y are distinguishable in \mathcal{X} . If there is both an $N \in \eta(x)$ such that $y \notin N$ and an $M \in \eta(y)$ such that $x \notin M$, then we call x and y T_1 -separated. Equivalently, in terms of closures, two distinct points x and y are distinguishable, if $x \notin \mathcal{C}(\{y\})$ or $y \notin \mathcal{C}(\{x\})$. They are T_1 -separated, if $(\{x\} \cap \mathcal{C}(\{y\})) \cup (\mathcal{C}(\{x\}) \cap \{y\}) = \emptyset$.*

The space \mathcal{X} is a symmetric space (or R_0 -space), if every two distinguishable points are T_1 -separated.

The following lemma implies that quasi-discrete paths that visit a non-quasi discrete point on a symmetric space cannot get back into “quasi-discrete territory”.

¹ To be exact, Kent and Min’s definition of neighbourhood spaces is more general than ours, as they do not require the neighbourhood systems to be filters. In fact, they show that a neighbourhood space where each neighbourhood system is a filter constitutes a pretopological space.



■ **Figure 1** Example of a topological path on a quasi-discrete space.

► **Lemma 8.** *Let $\mathcal{Q} = (Q, \eta_{\mathcal{Q}})$ be a quasi-discrete space and $\mathcal{X} = (X, \eta)$ be a non-quasi-discrete, but symmetric space. Furthermore let $x \in X$ be a point that does not have a minimal neighbourhood. Any continuous function $f: \mathcal{Q} \rightarrow \mathcal{X}$ that visits x at some point q can only visit points that are indistinguishable from x at any $q' \in N_{\min}(q)$. In terms of closures, this is equivalent to the following condition: if $q \in \mathcal{C}(\{q'\})$, then $f(q')$ is indistinguishable from x .*

Proof. Let $f: \mathcal{Q} \rightarrow \mathcal{X}$ be a continuous function with $f(q) = x$ and for some $q' \in N_{\min}(q)$, we have $f(q') = y$ where x and y are distinguishable. Hence, there is an $N \in \eta(x)$ such that $y \notin N$. However, for any $M \in \eta_{\mathcal{Q}}(q)$, we have that $N_{\min}(q) \subseteq M$, which of course means also $q' \in M$. But $f(q') \notin N$, so $f[M] \not\subseteq N$. So f is not continuous at q , which contradicts the assumption on f . ◀

We will often refer to the fact that quasi-discrete spaces closely resemble graphs: we can consider the points in the minimal neighbourhood of a point x to be connected to x by an edge. The following example provides a better understanding of the difference in behaviour of topological and quasi-discrete paths over quasi-discrete neighbourhood spaces.

► **Example 9.** Consider the quasi-discrete neighbourhood space \mathcal{X} in Fig. 1a. Any path p defined over $\mathcal{I}_{\mathbb{N}}$ is such that for any $i \in \mathcal{I}_{\mathbb{N}}$, if $p(i) = x$ or $p(i) = z$, then $p(j) = p(i)$ for any $j \geq i$. However, path p defined in Fig. 1b is a valid path when considering topological paths.

3 Spatial Logic for Neighbourhood Spaces

In this section, we briefly recall SLCS on general neighbourhood spaces. To that end, we first present spatial models based on neighbourhood spaces and then present the syntax and semantics of SLCS.

► **Definition 10** (Neighbourhood Model). *Let $\mathcal{X} = (X, \eta)$ be a neighbourhood space, \mathcal{I} an index space, AP a countable set of propositional atoms, and let $\nu: X \rightarrow \mathbb{P}(AP)$ be a valuation. Then $\mathcal{M} = (\mathcal{X}, \mathcal{I}, \nu)$ is a neighbourhood model over \mathcal{I} -paths. We will also write $\mathcal{M} = (X, \eta, \nu)$ to denote neighbourhood models, if the index space is clear from the context.*

We lift all suitable definitions from Sect. 2 to neighbourhood models in the obvious ways. For example, we will speak of continuous functions between the underlying spaces of two models as continuous functions between the models.

We will often use the special case of models with quasi-discrete spaces over quasi-discrete paths, since such models are graph-like models with standard paths on graphs.

► **Definition 11** (Purely Quasi-Discrete Models). *Let \mathcal{X} be a quasi-discrete neighbourhood space. A model $\mathcal{M} = (\mathcal{X}, \mathcal{I}_{\mathbb{N}}, \nu)$ over quasi-discrete paths is a purely quasi-discrete neighbourhood model.*

► **Definition 12** (Syntax of SLCS).

$$\varphi, \psi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathcal{N}\varphi \mid \varphi \mathcal{R}\psi \mid \varphi \mathcal{P}\psi$$

\mathcal{N} is read as near, \mathcal{R} is read as reachable from, and \mathcal{P} is read as propagates to.

The intuition behind the modalities is as follows. A point satisfies $\mathcal{N}\varphi$, if it is contained in the closure of the set of points satisfying φ . Hence, even if it does not satisfy φ itself, it is close to a point that does. A point x satisfies $\varphi \mathcal{R}\psi$ if there is a point y satisfying ψ such that x is reachable from y via a path where every point on this path between x and y satisfies φ . Propagation is in a sense the converse modality, i.e., if there is a point y satisfying ψ such that there is a path starting in x and reaching y at some index, and all points in between satisfy φ , then x satisfies $\varphi \mathcal{P}\psi$. This intuition is formalised in the following semantics.

► **Definition 13** (Path Semantics of SLCS). *Let $\mathcal{M} = ((X, \eta), \mathcal{I}, \nu)$ be a neighbourhood model and $x \in X$. The path semantics of SLCS with respect to \mathcal{M} are defined inductively as follows.*

$$\begin{aligned} \mathcal{M}, x \models p & \quad \text{iff } p \in \nu(x) \\ \mathcal{M}, x \models \neg\varphi & \quad \text{iff not } \mathcal{M}, x \models \varphi \\ \mathcal{M}, x \models \varphi \wedge \psi & \quad \text{iff } \mathcal{M}, x \models \varphi \text{ and } \mathcal{M}, x \models \psi \\ \mathcal{M}, x \models \mathcal{N}\varphi & \quad \text{iff } x \in \mathcal{C}(\{y \mid \mathcal{M}, y \models \varphi\}) \\ \mathcal{M}, x \models \varphi \mathcal{R}\psi & \quad \text{iff there is } p: y \rightsquigarrow \infty \text{ and } n \text{ such that } p(n) = x \text{ and } \mathcal{M}, y \models \psi \\ & \quad \text{and for all } 0 < i < n: \mathcal{M}, p(i) \models \varphi \\ \mathcal{M}, x \models \varphi \mathcal{P}\psi & \quad \text{iff there is } p: x \rightsquigarrow \infty \text{ and } n \text{ such that } \mathcal{M}, p(n) \models \psi \\ & \quad \text{and } \forall i: 0 < i < n \implies \mathcal{M}, p(i) \models \varphi \end{aligned}$$

In addition to the defined Boolean operators, we also allow for the other common derivable connectives. Specifically, $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$, $\top = \varphi \vee \neg\varphi$, $\perp = \neg\top$, $\varphi \rightarrow \psi = \neg\varphi \vee \psi$, and $\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. For a class of models \mathfrak{M} , we say that φ is *valid* in \mathfrak{M} if, and only if, $\mathcal{M}, x \models \varphi$ for every $\mathcal{M} = ((X, \eta), \mathcal{I}, \nu) \in \mathfrak{M}$ and $x \in X$.

► **Definition 14** (Relative Equivalence). *Let Σ be a subformula closed set of SLCS formulas, \mathcal{M} a neighbourhood model, and $x, y \in \mathcal{M}$ be two points of \mathcal{M} . Then x and y are equivalent relative to Σ iff they satisfy the same formulas in Σ , i.e., $x \simeq_{\Sigma} y$ iff $\{\varphi \in \Sigma \mid \mathcal{M}, x \models \varphi\} = \{\varphi \in \Sigma \mid \mathcal{M}, y \models \varphi\}$. This is an equivalence relation, and we will denote the equivalence classes of x by $[x]_{\Sigma}$ and $[x]$, if Σ is clear from the context.*

The following lemmas present properties of formulas on different classes of models. We start with the most familiar class: purely quasi-discrete models. On these models, we have a clear connection between the near modality and the propagate path operator.

► **Lemma 15.** *On all purely quasi-discrete neighbourhood models $\mathcal{M} = (\mathcal{X}, \mathcal{I}_{\mathbb{N}}, \nu)$ we have that $\mathcal{M}, x \models \mathcal{N}\varphi$ iff $\mathcal{M}, x \models \varphi \vee \perp \mathcal{P}\varphi$.*

Proof. If $\mathcal{M}, x \models \varphi$, the equivalence is clear. Otherwise, assume $\mathcal{M}, x \models \perp \mathcal{P}\varphi$. This means that there is a point y and a path $p: x \rightsquigarrow \infty$ such that $p(1) = y$ and $\mathcal{M}, y \models \varphi$. Since p is continuous, this means that there is a neighbourhood N of 0 such that $p[N] \subseteq N_{\min}(x)$. Since every neighbourhood of 0 contains 1, this means $y \in N_{\min}(x)$, and so $\mathcal{M}, x \models \mathcal{N}\varphi$. The other direction is similar. ◀

If we consider quasi-discrete models over topological paths, this connection is less clear. The main reason for this is that over topological graphs, $\perp \mathcal{P} \varphi$ is equivalent to φ , which is easy to prove. However, we can still establish a bit less obvious connection between the modalities.

► **Lemma 16.** *On quasi-discrete models over topological paths, $(a \wedge \mathcal{N}(b \wedge \neg a)) \rightarrow \mathcal{N}(\neg a \wedge (b \mathcal{P} a))$ is valid.*

Proof. Let $\mathcal{M} = (\mathcal{X}, \mathcal{I}_{\mathbb{R}}, \nu)$ with $\mathcal{X} = (X, \eta)$ be a quasi-discrete model and let $x \in X$ such that $\mathcal{M}, x \models a \wedge \mathcal{N}(b \wedge \neg a)$. That is, $x \models a$ and $x \in \mathcal{C}(\{y \mid \mathcal{M}, y \models b \wedge \neg a\})$. Since \mathcal{X} is quasi-discrete, this means that there is a $y \in N_{min}(x)$ such that $\mathcal{M}, y \models b \wedge \neg a$. Then, the path $p: \mathcal{I}_{\mathbb{R}} \rightarrow \mathcal{X}$ with $p(i) = y$ for $i < 1$ and $p(i) = x$ for $i = 1$ is a witness for $\mathcal{M}, y \models b \mathcal{P} a$. This function is indeed continuous: Consider $N \in \eta(p(i))$. If $i < 1$, we can always choose an $N_i \in \eta_{\mathcal{I}}(i)$ such that $\forall j \in N_i$ we have $j < 1$, since \mathcal{I} has arbitrarily small neighbourhoods, which means $p[N_i] = \{y\} \subseteq N$. If $i = 1$, we have for any neighbourhood $N_i \in \eta_{\mathcal{I}}(i)$, that is $p[N_i] \subseteq \{x, y\} \subseteq N_{min}(x) \subseteq N$. Furthermore, $p(0) = y$, and for $n = 1$, we have $p(n) = x$, and for all $0 < i < n$, $\mathcal{M}, p(i) \models b$. Since $y \in N_{min}(x)$, we have that $\mathcal{M}, x \models \mathcal{N}(\neg a \wedge (b \mathcal{P} a))$. ◀

Furthermore, on any kind of model over topological paths, we get that the reachable and propagate modalities are equivalent. Intuitively, this is clear, since for topological paths, there is no inherent direction on the index space, in contrast to the quasi-discrete index space, where the successor relation is directed.

► **Lemma 17.** *On any neighbourhood model over topological paths $\mathcal{M} = (\mathcal{X}, \mathcal{I}_{\mathbb{R}}, \nu)$ we have that $\mathcal{M}, x \models \varphi \mathcal{P} \psi$ iff $\mathcal{M}, x \models \varphi \mathcal{R} \psi$.*

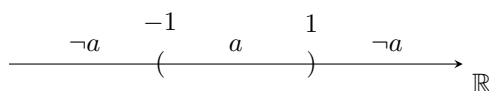
Proof. Let $\mathcal{M} = ((X, \eta), \mathcal{I}_{\mathbb{R}}, \nu)$ be a neighbourhood model over topological paths, and $x \in X$ a point of \mathcal{M} such that $\mathcal{M}, x \models \varphi \mathcal{P} \psi$. So there is a path $p: \mathcal{I}_{\mathbb{R}} \rightarrow \mathcal{M}$ and $n \in [0, 1]$, such that $p(0) = x$, $p(n) = y$ and $\mathcal{M}, y \models \psi$, and $\forall k: 0 < k < n$, we have $\mathcal{M}, p(k) \models \varphi$. Since p is topological, we can assume without loss of generality that $n = 1$. Now the path p' defined by $p'(i) = p(1 - i)$ is a witness for $\mathcal{M}, x \models \varphi \mathcal{R} \psi$. Indeed, let $N \in \eta(p'(i))$ be a neighbourhood of $p'(i)$. By definition of p' , we have $p'(i) = p(1 - i)$. We know that p is continuous at $1 - i$, so there is a neighbourhood $N' \in \eta_i(1 - i)$ such that $p[N'] \subseteq N$. But, we also have that $N^i = \{j \mid 1 - j \in N'\}$ is a neighbourhood of i and, since $p'(j) = p(1 - j)$, we have that $p'[N^i] \subseteq N$ as well. So, p' is continuous. Furthermore, $p'(0) = p(1)$, so $\mathcal{M}, p'(0) \models \psi$, $p'(1) = x$, and for all k with $0 < k < 1$, we have $\mathcal{M}, p'(k) \models \varphi$, by definition of p' . The other direction is similar. ◀

4 No Finite Model Property for Arbitrary Neighbourhood Spaces

In this section, we prove that SLCS does not have the finite model property if we consider the class of all neighbourhood models. That is, we show that there exist SLCS formulas that are satisfiable only over models $\mathcal{M} = ((X, \eta), \mathcal{I}, \nu)$ where X is not finite. Our first observation is that there are satisfiable formulas that are not satisfiable on purely quasi-discrete models.

► **Lemma 18.** *There exist SLCS satisfiable formulas that are not satisfiable on any finite model over quasi-discrete paths.*

Proof. Consider model $\mathcal{M} = ((\mathbb{R}, \eta_{\mathbb{R}}), \mathcal{I}_{\mathbb{R}}, \nu)$ in Fig. 2. It follows that $\mathcal{M}, 1 \models \mathcal{N} a \wedge \neg a \wedge \neg(\perp \mathcal{P} a)$. By Lemma 15, this formula is a contradiction on purely quasi-discrete models. Finally, since every finite space is quasi-discrete, the lemma holds. ◀



■ **Figure 2** Model $\mathcal{M} = ((\mathbb{R}, \eta_{\mathbb{R}}), \mathcal{I}_{\mathbb{R}}, \nu)$ such that $\mathcal{M}, 1 \models \mathcal{N}a \wedge \neg a \wedge \neg(\perp \mathcal{P}a)$.

There are two key differences between the model in Fig. 2 and purely quasi-discrete models: the type of underlying space, and the type of paths allowed. So, we now restrict both of these dimensions one after the other. First, we show that SLCS does not admit finite models over topological paths, if we consider the full set of neighbourhood spaces, by constructing a counterexample based on the result of Lemma 16.

► **Lemma 19.** *There exist SLCS formulas that are satisfiable on models with topological paths, but not on any finite model with topological paths.*

Proof. We construct a topological model $\mathcal{M} = (\mathcal{X}, \mathcal{I}_{\mathbb{R}}, \nu)$ that contains a point satisfying $a \wedge \mathcal{N}(b \wedge \neg a) \wedge \neg \mathcal{N}(\neg a \wedge (b \mathcal{P}a))$. For the topological space, we use the *topologists sine curve*. For that purpose, let $S = \{(r, \sin \frac{1}{r}) \mid 0 < r \leq 1\}$. The space is then defined by $\mathcal{X} = (X, \eta)$, where $X = \{(0, 0)\} \cup S$, and η is the neighbourhood system induced by treating this set as a subset of the Euclidean plane \mathbb{R}^2 . That is, $N \in \eta(x)$ if there is an open ball of some radius r around x , i.e., some $B_r = \{y \mid \|x - y\| < r\}$, where $\|\cdot\|$ is the Euclidean distance, such that $N \supseteq B_r \cap X$. We set the valuation ν by $\nu((0, 0)) = \{a\}$ and $\nu(x) = \{b\}$ for $x \neq (0, 0)$.

Now, every neighbourhood of $(0, 0)$ contains a value from S , and thus $\mathcal{M}, (0, 0) \models a \wedge \mathcal{N}(b \wedge \neg a)$. Furthermore, it is well known [21] that in this space, $(0, 0)$ is not path-connected to S , which means that no path starting in any point $s \in S$ can reach $(0, 0)$. This implies, that no point $s \in S$ satisfies $b \mathcal{P}a$, since there is no path that ever reaches a point that satisfies a . So, no point on the model satisfies $\neg a \wedge (b \mathcal{P}a)$. In particular, this means that $\mathcal{M}, (0, 0) \models \neg \mathcal{N}(\neg a \wedge (b \mathcal{P}a))$. So, we have $\mathcal{M}, (0, 0) \models a \wedge \mathcal{N}(b \wedge \neg a) \wedge \neg \mathcal{N}(\neg a \wedge (b \mathcal{P}a))$. But this formula is not satisfiable on any quasi-discrete model with topological paths, according to Lemma 16. Since finite models are quasi-discrete, SLCS does not generally admit finite models over topological paths. ◀

Finally, even when considering only quasi-discrete paths, there are SLCS formulas which are not satisfiable on finite models.

► **Lemma 20.** *There exist SLCS formulas that are satisfiable on models with quasi-discrete paths, but not on any finite model with quasi-discrete paths.*

Proof. Let X be an infinite, uncountable set and let $\mathcal{X} = (X', \eta)$ be the double pointed countable complement topology over X (see [21]). For this definition, let \mathcal{Y} be the set of all subsets of X , such that for every $Y \in \mathcal{Y}$, either $Y = \emptyset$, or the complement of Y is countable. X' is constructed from X by “doubling” all points, i.e., $X' = \{x' \mid x \in X\} \cup X$, where each x' is a new, distinct, element to the x it is constructed from. Then, let \mathcal{Y}' be the doubling of every set in \mathcal{Y} in a similar way, and η be defined by $\eta(x) = \{N \mid \exists Y \in \mathcal{Y}' : Y \subseteq N \wedge x \in Y\}$. Note that this definition implies that for any y and its doubled point y' , we have $\eta(y) = \eta(y')$. Define $\mathcal{M} = (\mathcal{X}, \mathcal{I}_{\mathbb{N}}, \nu)$ by letting $x, x' \in X'$ be a designated pair of points in X' and ν be given by $\nu(y) = \{a\}$, if $y \in \{x, x'\}$ and $\nu(y) = \{b\}$ otherwise.

Now consider any neighbourhood $N \in \eta(x)$. There is always some $y \in N$ that is different from x and x' , since otherwise the complement of N would be uncountable. Hence, every neighbourhood N contains some element y with $\mathcal{M}, y \models b$, which implies $\mathcal{M}, x \models \mathcal{N}b$. However, since the underlying space of \mathcal{M} is symmetric, by Lemma 8, any quasi-discrete

path starting in x may only visit x or x' , which both do not satisfy b . Hence $\mathcal{M}, x \not\models \perp \mathcal{P} b$. So, $\mathcal{N} b \wedge \neg(\perp \mathcal{P} b)$ is satisfiable on this model. But no finite model can satisfy this formula, since it is necessarily purely quasi-discrete. \blacktriangleleft

5 Finite Model Property for Quasi-Discrete Spaces

In this section, we prove that SLCS admits finite models if we restrict the class of models to quasi-discrete models. That is, the models correspond to directed graphs. Our approach is similar to standard approaches in modal logic [4]. In particular, we use filtrations with respect to a subformula closed set Σ for both types of models. Since topological paths and quasi-discrete paths behave very differently, we further distinguish the class into models over quasi-discrete paths and over topological paths.

5.1 Quasi-Discrete Spaces with Quasi-Discrete Paths

In this subsection, we prove that SLCS has the finite model property on purely quasi-discrete neighbourhood models. That is, the paths are similar to typical paths on graph structures.

The following lemma allow us to transfer information about the satisfaction of the path operators to other points.

► **Lemma 21.** *Let \mathcal{M} be a purely quasi-discrete neighbourhood model and $x, y \in \mathcal{M}$ two points such that $y \in N_{min}(x)$. Then the following hold.*

1. *If $\mathcal{M}, y \models \varphi$ and $\mathcal{M}, y \models \varphi \mathcal{P} \psi$, then also $\mathcal{M}, x \models \varphi \mathcal{P} \psi$.*
2. *If $\mathcal{M}, x \models \varphi \mathcal{R} \psi$ and $\mathcal{M}, x \models \varphi$, then also $\mathcal{M}, y \models \varphi \mathcal{R} \psi$.*

Proof. We only prove the first statement as the second is similar.

From $\mathcal{M}, y \models \varphi \mathcal{P} \psi$ we know that there is a path $p: \mathcal{I} \rightarrow \mathcal{M}$ with $p(0) = y$ and an index $n \in \mathcal{I}$ such that $\mathcal{M}, p(n) \models \psi$ and for all $0 < i < n$, we have $\mathcal{M}, p(i) \models \varphi$. Now consider the continuous function $p_x: \mathcal{I} \rightarrow \mathcal{M}$ given by $p_x(0) = x$ and $p_x(i+1) = p(i)$. Then p_x is indeed a path, since \mathcal{M} is quasi-discrete and $y \in N_{min}(x)$. Also, we have $\mathcal{M}, p_x(n+1) \models \psi$ and, since $\mathcal{M}, y \models \varphi$, for all $0 < i < n+1$, we have $\mathcal{M}, p_x(i) \models \varphi$. Hence $\mathcal{M}, x \models \varphi \mathcal{P} \psi$. \blacktriangleleft

We now define filtrations for purely quasi-discrete models. Most parts of this definition are standard, when we consider \mathcal{N} similar to an existential modality. For the two path operators, we added additional properties that allow us to transfer information about the existence of paths from the filtration back to the original model.

► **Definition 22 (Filtration).** *Let Σ be a subformula closed set of SLCS formulas, and $\mathcal{M} = (X, \eta, \nu)$ a purely quasi-discrete neighbourhood model. We call a purely quasi-discrete neighbourhood model $\mathcal{M}_f = (X_f, \eta_f, \nu_f)$ a filtration of \mathcal{M} through Σ , if it satisfies the following conditions:*

1. $X_f = \{[x]_\Sigma \mid x \in X\}$
2. *if $y \in N_{min}(x)$, then $[y] \in N_{min}([x])$*
3. *if $[y] \in N_{min}([x])$, then for each $\mathcal{N}\varphi \in \Sigma$, we have that if $\mathcal{M}, y \models \varphi$, then $\mathcal{M}, x \models \mathcal{N}\varphi$*
4. *if there is a sequence $[x_0] \dots [x_n]$ with $[x_{i+1}] \in N_{min}([x_i])$ for all $0 \leq i < n$, then for every $\varphi \mathcal{P} \psi \in \Sigma$, we have that whenever $\mathcal{M}, x_i \models \varphi$ for each $0 < i < n$ and $\mathcal{M}, x_n \models \psi$, then also $\mathcal{M}, x_0 \models \varphi \mathcal{P} \psi$*
5. *if there is a sequence $[x_0] \dots [x_n]$ with $[x_{i+1}] \in N_{min}([x_i])$ for all $0 \leq i < n$, then for every $\varphi \mathcal{R} \psi \in \Sigma$, we have that whenever $\mathcal{M}, x_i \models \varphi$ for each $0 < i < n$ and $\mathcal{M}, x_n \models \psi$, then also $\mathcal{M}, x_0 \models \varphi \mathcal{R} \psi$*
6. $\nu_f([x]) = \{p \in AP \mid \mathcal{M}, x \models p\}$

As usual, satisfiability of formulas in Σ is preserved between a model and its filtration through Σ . So our filtration is properly defined.

► **Lemma 23.** *Let \mathcal{M}_f be a filtration of \mathcal{M} through Σ . Then for all $\varphi \in \Sigma$, we have $\mathcal{M}, x \models \varphi$ iff $\mathcal{M}_f, [x] \models \varphi$.*

Proof. We proceed by induction on the structure of formulas. The base case for atomic propositions is immediate by Def. 22. The cases for the boolean operators are standard.

The case for $\varphi = \mathcal{N}\psi$ is similar to standard modal logic [4]: we have $\mathcal{M}, x \models \mathcal{N}\psi$ iff $x \in \mathcal{C}(\{y \mid \mathcal{M}, y \models \psi\})$ which by definition of the closure is equivalent to $\forall N \in \eta(x): N \cap \{y \mid \mathcal{M}, y \models \psi\} \neq \emptyset$. On quasi-discrete models, this is equivalent to $\exists y \in N_{\min}(x): \mathcal{M}, y \models \psi$. By property 2 of filtrations and the induction hypothesis, this implies $\exists [y] \in N_{\min}([x]): \mathcal{M}_f, [y] \models \psi$. Applying similar equivalences as before, we get that $\mathcal{M}_f, [x] \models \mathcal{N}\psi$. Conversely, assume we have $\mathcal{M}_f, [x] \models \mathcal{N}\psi$. With the same reasoning as above, this is equivalent to $\exists [y] \in N_{\min}([x]): \mathcal{M}_f, [y] \models \psi$. By the induction hypothesis, we get $\mathcal{M}, y \models \psi$, and from property 3 of filtrations, we have $\mathcal{M}, x \models \mathcal{N}\psi$.

Now consider $\varphi = \psi \mathcal{P} \chi$. If $\mathcal{M}, x \models \psi \mathcal{P} \chi$, this is equivalent to the existence of a path $p: x \rightsquigarrow \infty$ and a n and $\mathcal{M}, p(n) \models \chi$ as well as $\forall i: 0 < i < n$, we have $\mathcal{M}, p(i) \models \psi$. That is, there is a sequence x_0, \dots, x_n such that $x_0 = x$ and $x_{i+1} \in N_{\min}(x_i)$ for all $i < n$. By property 2, we have $[x_{i+1}] \in N_{\min}([x_i])$ for all $i < n$, and by the induction hypothesis, $\mathcal{M}_f, [x_n] \models \chi$ and for all $0 < i < n$, we get $\mathcal{M}_f, [x_i] \models \psi$. That is, $\mathcal{M}_f, [x] \models \psi \mathcal{P} \chi$. Conversely, assume $\mathcal{M}_f, [x] \models \psi \mathcal{P} \chi$. Then there is a sequence $[x_0], \dots, [x_n]$ such that $[x_{i+1}] \in N_{\min}([x_i])$ for all $0 \leq i < n$, and $\mathcal{M}_f, [x_n] \models \chi$, as well as for all $0 < i < n$, we get $\mathcal{M}_f, [x_i] \models \psi$. By the induction hypothesis, we get $\mathcal{M}, x_n \models \chi$ and $\mathcal{M}, x_i \models \psi$ for every $0 < i < n$. Hence, by property 4, and since $x_0 \simeq x$, we have $\mathcal{M}, x \models \psi \mathcal{P} \chi$.

The case for $\psi \mathcal{R} \chi$ is similar, by using property 5. ◀

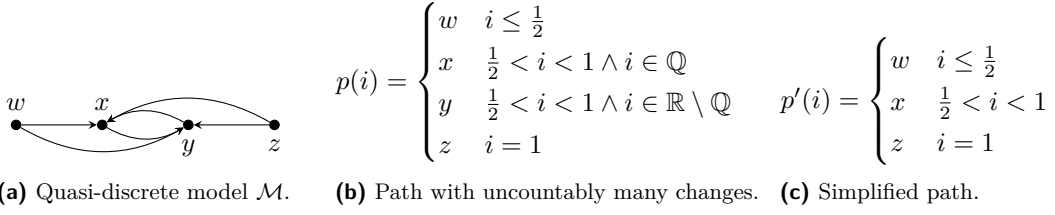
Finally, we prove that there is always a filtration through Σ for any given purely quasi-discrete model. This definition corresponds to the usual definition of smallest filtration [4].

► **Lemma 24.** *Let Σ be a subformula closed set of formulas and \mathcal{M} a purely quasi-discrete model. Furthermore, let X_Σ be the set of equivalence classes of \simeq_Σ , ν_Σ be defined as in Def. 22 (6), and $\eta_s([x]) = \langle \{[y] \mid \exists y', x': y' \in [y] \wedge x' \in [x] \wedge y' \in N_{\min}(x')\} \rangle$ for each $[x] \in X_\Sigma$. Then the model $(X_\Sigma, \eta_s, \nu_\Sigma)$ is a filtration of \mathcal{M} through Σ .*

Proof. Properties 1, 2 and 6 are immediate. So now assume that $[y] \in N_{\min}([x])$ and let $\mathcal{N}\varphi \in \Sigma$ such that $\mathcal{M}, y \models \varphi$. Then by definition of η_s , there are $x' \in [x]$ and $y' \in [y]$ such that $y' \in N_{\min}(x')$. Since $y \simeq_\Sigma y'$, we have $\mathcal{M}, y' \models \varphi$, and due to $y' \in N_{\min}(x')$, this implies $x' \in \mathcal{C}(\{y \mid \mathcal{M}, y \models \varphi\})$, which means $\mathcal{M}, x' \models \mathcal{N}\varphi$. Since $x \simeq_\Sigma x'$, this implies $\mathcal{M}, x \models \mathcal{N}\varphi$. Hence property 3 holds.

For proving property 4, we proceed by induction on the length of sequence $[x_0] \dots [x_n]$. For the base case, we have $\mathcal{M}, x_0 \models \psi$, which implies $\mathcal{M}, x_0 \models \varphi \mathcal{P} \psi$. So, assuming the property holds for suited sequences of length up to n , consider a sequence $[x_0] \dots [x_n]$ such that the conditions of the property are satisfied. In particular, $[x_1] \dots [x_n]$ is a sequence, where $[x_{i+1}] \in N_{\min}([x_i])$, and for all $1 < i < n$ we have $\mathcal{M}, x_i \models \varphi$ and $\mathcal{M}, x_n \models \psi$. Hence, by the induction hypothesis, $\mathcal{M}, x_1 \models \varphi \mathcal{P} \psi$. Furthermore, by assumption on the sequence, we get $\mathcal{M}, x_1 \models \varphi$. Now, by the definition of η_s , we know that there are $x'_0 \in [x_0]$ and $x'_1 \in [x_1]$ such that $x'_1 \in N_{\min}(x'_0)$, and since $x_1 \simeq x'_1$, both $\mathcal{M}, x'_1 \models \varphi$ as well as $\mathcal{M}, x'_1 \models \varphi \mathcal{P} \psi$ hold. Hence, by Lemma 21 (1), we have $\mathcal{M}, x'_0 \models \varphi \mathcal{P} \psi$, and since $x_0 \simeq x'_0$, also $\mathcal{M}, x_0 \models \varphi \mathcal{P} \psi$.

Property 5 can be proven similarly to the previous case, but using Lemma 21 (2). ◀



■ **Figure 3** Example of path simplification.

From the definition of filtration and Lemmas 23 and 24, where X_Σ is finite as the set of subformulas of a formula is finite, we obtain our first finite model property result.

► **Theorem 25.** *If φ is a SLCS formula that is satisfiable on a purely quasi-discrete neighbourhood model, then φ is satisfiable on a finite purely quasi-discrete neighbourhood model.*

5.2 Quasi-Discrete Spaces with Topological Paths

In this section, we prove that SLCS also admits finite models for the class of quasi-discrete models over topological paths. This case is interesting, since topological paths behave very differently from quasi-discrete paths. For example, topological paths are not required to comply with the direction of the edges of the underlying graph.

► **Example 26.** Consider the model in Fig. 3a. We can define a topological path p as in Fig. 3b. This function is indeed continuous. For $i < \frac{1}{2}$, the function is constant, since it is constant. At $i = \frac{1}{2}$, we have that for the minimal neighbourhood $N_{\min}(w) = \{w, x, y\}$, we can always find a neighbourhood N' of $\frac{1}{2}$ that does not contain 1, and so $p[N'] \subseteq N_{\min}(w)$. If $\frac{1}{2} < i < 1$, then $N_{\min}(p(i)) = \{x, y\}$, and we can choose any neighbourhood $N' \in \eta(i)$ that does not contain values less than $\frac{1}{2}$ and greater or equal to 1 to show continuity. At 1, the function is continuous for similar reasons as at $\frac{1}{2}$. So the function is a path.

However, path p contains many “superfluous detours” in the set $\{x, y\}$. A simpler path would be path p' in Fig. 3c, or a variation in which p' maps to y instead of x . This path only visits points that were visited by p as well, but omits these detours.

The following Lemma formalises the intuition explained in Example 26. We will use it to normalise the paths used as witnesses for the satisfaction of the propagate modality when we prove the existence of filtrations.

► **Remark 27.** From this point onward, we will use the following slight abuse of notation. For two indices $r, s \in [0, 1]$, we write $p[r, s] = \{p(i) \mid r < i < s\}$ to denote the values of a path p on the open interval between r and s . If $p[r, s]$ is a singleton (i.e., p is constant on the interval (r, s)), we will also treat $p[r, s]$ as a single value, to avoid unnecessary parentheses.

► **Lemma 28 (Path Simplification).** *Let $\mathcal{M} = ((X, \eta), \mathcal{I}_{\mathbb{R}}, \nu)$ a neighbourhood model, where (X, η) is a quasi-discrete space, and let $p: [0, 1] \rightarrow X$ be a path on \mathcal{M} such that p has a finite image. Then there is a path p' and a sequence of indices i_0, \dots, i_n with $i_0 = 0$, $i_n = 1$ and $i_r < i_{r+1}$ for all $r < n$, such that*

1. $p'(i) = p(i)$ for all the indices in the sequence,
2. p' is constant on each open interval (i_r, i_{r+1}) ,
3. $p'[i_r, i_{r+1}] \neq p'[i_s, i_{s+1}]$ for $r \neq s$,
4. if $p'(i_{r+1}) \neq p'[i_r, i_{r+1}]$, then $p'[i_r, i_{r+1}] \in N_{\min}(p'(i_{r+1}))$,
5. if $p'(i_r) \neq p'[i_r, i_{r+1}]$, then $p'[i_r, i_{r+1}] \in N_{\min}(p'(i_r))$,
6. if $p(i) \neq p'(i)$, then there are $r, s \in [0, 1]$ and $y \in X$ with $r < i < s$ such that $p(r) = p(s) = y$ and $p'(r) = p'(s) = y$.

Proof. Let \mathcal{M} and p be as required, let $x \in X$ be a point in the space, and $0 \leq s \leq 1$ an index. We indicate by $\text{sI}(p, x, s)$ the smallest subinterval I of $[s, 1]$ such that $\forall i \in [s, 1] \setminus I$ it holds that $p(i) \neq x$. Let a be the infimum (resp., supremum) of $\text{sI}(p, x, s)$, then it follows that $\forall N \in \eta(a)$ there exists an $i \in N \cap \text{sI}(p, x, s)$ such that $p(i) = x$.

We now construct the sequence of indices i_0, \dots, i_n and the path p' . We set $i_0 = 0$, $p'(0) = p(0)$, and then proceed as follows starting from $\text{sI}(p, p(0), i_0)$.

Consider an index i_k , a point $x \in X$, and let a be the supremum of $\text{sI}(p, x, i_k)$. We set $p'(i) = x$ for all $i_k < i < a$, we set $p'(a) = p(a)$, and

1. if $a \notin \text{sI}(p, x, i_k)$, we set $i_{k+1} = a$, and then proceed with $\text{sI}(p, p(a), i_{k+1})$;
2. otherwise (i.e., $a \in \text{sI}(p, x, i_k)$), we need to find a possible way to proceed with the path following the index a . That is, we need to find the right point and index for the function sI . Let $S = \{y \in N_{\min}(p(a)) \mid \forall N \in \eta(a): y \in p[N \cap [a, 1]]\} \setminus \{p(a)\}$. Observe that $S \neq \emptyset$ as p is a continuous function on X , and any point in S is a good candidate for the continuation of the construction. Now we need to understand whether or not to move from the index i_k to the index i_{k+1} . If $i_k = a$, then we proceed by choosing any of the $y \in S$ and considering $\text{sI}(p, y, i_k)$. Otherwise, we proceed by choosing any of the $y \in S$, setting $i_{k+1} = a$, and considering $\text{sI}(p, y, i_{k+1})$.

Since p has a finite image, the process above terminates when $i_k = 1$.

Now let p' be the path constructed as above. Properties 1, 2 and 3 are immediate results of the construction of p' . Let us show that property 4 holds, and consider the case where $p'(i_{r+1}) \neq p'[i_r, i_{r+1}]$. By construction we know that i_{r+1} is the supremum of $\text{sI}(p, x, i_r)$, which means that $\forall N \in \eta(i_{r+1}) \exists i \in N \cap (i_r, i_{r+1})$ with $p(i) = x = p'[i_r, i_{r+1}]$. By continuity of p it must hold that $\exists N' \in \eta(i_{r+1})$ such that $p[N'] \subseteq N_{\min}(p(i_{r+1}))$. As $p'[i_r, i_{r+1}] \in p[N']$, then $p'[i_r, i_{r+1}] \in N_{\min}(p'(i_{r+1}))$. Property 5 follows immediately from point 2 above since we select y among the elements in the minimal neighbourhood. Finally we consider property 6. Let i be an index such that $p(i) \neq p'(i)$. By property 1, we know that i cannot be any of the indices in the resulting sequence. Let i_k and i_{k+1} be the two indices in the resulting sequence such that $i_k < i < i_{k+1}$. By definition of $\text{sI}(p, p'(i), i_k)$, there must exist two indices r and s such that $p(r) = p(s) = p'(i)$, and $i_k \leq r < i < s \leq i_{k+1}$. By property 2 $p'[i_k, i_{k+1}] = p'(i)$, and the property holds. \blacktriangleleft

Similarly to the case with quasi-discrete paths, the following lemma allow us to transfer information about the satisfaction of the path operator to neighbouring points.

► **Lemma 29.** *Let \mathcal{M} be a quasi-discrete neighbourhood model over topological paths and $x, y \in \mathcal{M}$ two points. Then the following hold.*

1. *If $y \in N_{\min}(x)$, $\mathcal{M}, y \models \varphi$ and $\mathcal{M}, y \models \varphi \mathcal{P} \psi$, then also $\mathcal{M}, x \models \varphi \mathcal{P} \psi$.*
2. *If $x \in N_{\min}(y)$, $\mathcal{M}, x \models \varphi$, $\mathcal{M}, y \models \varphi$ and $\mathcal{M}, y \models \varphi \mathcal{P} \psi$, then also $\mathcal{M}, x \models \varphi \mathcal{P} \psi$.*

Proof. Case (1): Let p and n be witnesses for $\mathcal{M}, y \models \varphi \mathcal{P} \psi$. There are two cases to consider. In the first case, p stays on y for an infinite number of indices. That is, the initial segment of p is not a singleton. Then we can define p' by $p'(0) = x$ and $p'(i) = p(i)$ for $i > 0$. Since p is continuous p' is continuous for every $i > 0$. For $i = 0$, we can take any neighbourhood $N \in \eta_{\mathbb{R}}(0)$ that only extends into the initial segment of p , where $p(j) = y$ for any $i \in N$ with $i \neq 0$. Then $p'[N] \subseteq N_{\min}(x)$. So p' is also continuous at 0, and since $\mathcal{M}, y \models \varphi$, it is a witness for $\mathcal{M}, x \models \varphi \mathcal{P} \psi$. In the other case, p stays on y for the single index 0, and then moves to some point z . Then we define p' by $p'(0) = x$, $p'(i) = y$ for $0 < i \leq \frac{1}{2}$ and $p'(i) = p(2i - 1)$ for $i > \frac{1}{2}$. Similar to the case above, p' is continuous at 0. Since the constant path is continuous, p' is continuous at $0 < i < \frac{1}{2}$. And since p is continuous at $2i - 1$, p' is continuous at i for $i \geq \frac{1}{2}$. Furthermore, with $n' = \frac{1}{2}(n + 1)$, p' is a witness for $\mathcal{M}, x \models \varphi \mathcal{P} \psi$.

Case (2): By assumption on y , there is a path $p: \mathbb{R} \rightarrow \mathcal{M}$ and a value n , such that $p(0) = y$, $\mathcal{M}, p(n) \models \psi$ and for all i with $0 < i < n$, we have $\mathcal{M}, p(i) \models \varphi$. Using this path, we can construct the path p' by setting $p'(i) = x$ if $i < \frac{1}{2}$ and $p'(i) = p(2i - 1)$ for $i \geq \frac{1}{2}$. This function is continuous, and thus a path. Furthermore, we have $\mathcal{M}, p'(n+1) \models \psi$, and of course for all i with $0 < i < \frac{1}{2}(n+1)$ we have $\mathcal{M}, p'(i) \models \varphi$. So this path is a witness for $\mathcal{M}, x \models \varphi \mathcal{P} \psi$. \blacktriangleleft

We now proceed with the definition of filtrations for quasi-discrete models over topological paths. As can be expected, the definition differs from Def. 22 only in the treatment of paths. Instead of explicitly enumerating the equivalence classes on a path, we only assume the existence of a path on the filtration, and then transfer the satisfaction back to the original model. Furthermore, we do not need to consider the reachability path operator, since it is equivalent to the propagate modality, by Lemma 17.

► **Definition 30** (Filtration with Topological Paths). *Let Σ be a subformula closed set of SLCS formulas, and $\mathcal{M} = ((X, \eta), \mathcal{I}_{\mathbb{R}}, \nu)$ a neighbourhood model, where (X, η) is a quasi-discrete space. We call the neighbourhood model $\mathcal{M}_f = ((X_f, \eta_f), \mathcal{I}_{\mathbb{R}}, \nu_f)$ a filtration of \mathcal{M} over topological paths through Σ , if it satisfies the following conditions:*

1. $X_f = \{[x]_{\Sigma} \mid x \in X\}$
2. if $y \in N_{\min}(x)$, then $[y] \in N_{\min}([x])$
3. if $[y] \in N_{\min}([x])$, then for each $\mathcal{N} \varphi \in \Sigma$, we have that if $\mathcal{M}, y \models \varphi$, then $\mathcal{M}, x \models \mathcal{N} \varphi$
4. if $\pi: [0, 1] \rightarrow X_f$ is a path on \mathcal{M}_f where $\pi(i) = [x_i]$, then for every $\varphi \mathcal{P} \psi \in \Sigma$, we have that whenever $\mathcal{M}, x_i \models \varphi$ for each $0 < i < n$ and $\mathcal{M}, x_n \models \psi$, then also $\mathcal{M}, x_0 \models \varphi \mathcal{P} \psi$
5. $\nu_f([x]) = \{p \in AP \mid \mathcal{M}, x \models p\}$

As in the purely quasi-discrete case, satisfaction of all formulas in the subformula closed set Σ is preserved on filtrations through Σ .

► **Lemma 31.** *Let \mathcal{M}_f be a filtration of the quasi-discrete model \mathcal{M} over topological paths through Σ . Then for all $\varphi \in \Sigma$, we have $\mathcal{M}, x \models \varphi$ iff $\mathcal{M}_f, [x] \models \varphi$.*

Proof. We proceed by induction on the structure of formulas. The base case for atomic propositions is immediate by Def. 30. The cases for the boolean operators are standard and the case for $\varphi = \mathcal{N} \psi$ is exactly as for Lemma 23.

Now consider $\varphi = \psi \mathcal{P} \chi$. If $\mathcal{M}, x \models \psi \mathcal{P} \chi$, this is equivalent to the existence of a path $p: x \rightsquigarrow \infty$ and a n and $\mathcal{M}, p(n) \models \chi$ as well as $\forall i: 0 < i < n$, we have $\mathcal{M}, p(i) \models \psi$. Observe that for any j and k such that $p(k) \in N_{\min}(p(j))$, we have $[p(k)] \in N_{\min}([p(j)])$ by property 2. Furthermore, for any j , we know that there is a $N \in \eta(j)$ such that $p[N] \subseteq N_{\min}(p(j))$ by continuity of p . So, these two facts together imply that $\forall k \in N$, we have $[p(k)] \in N_{\min}([p(j)])$. Hence we can define $\pi: [0, 1] \rightarrow X_f$ by $\pi(i) = [p(i)]$ and then have that π is a path on \mathcal{M}_f such that $\pi(0) = [x]$. Furthermore, by the induction hypothesis, for all i with $0 < i < n$, we have $\mathcal{M}_f, \pi(i) \models \psi$ and $\mathcal{M}_f, \pi(n) \models \chi$. This of course means $\mathcal{M}_f, [x] \models \psi \mathcal{P} \chi$.

Conversely, assume $\mathcal{M}_f, [x] \models \psi \mathcal{P} \chi$. Then there is a path $\pi: [0, 1] \rightarrow X_f$ such that $\pi(0) = [x]$, for all i with $0 < i < n$ we have $\mathcal{M}_f, \pi(i) \models \psi$ and $\mathcal{M}_f, \pi(n) \models \chi$. Let $\pi(i) = [x_i]$, then we get by the induction hypothesis that $\mathcal{M}, x_i \models \psi$ for all i with $0 < i < n$ and $\mathcal{M}, x_n \models \chi$. By property 4 we get $\mathcal{M}, x_0 \models \psi \mathcal{P} \chi$ and by $x \simeq x_0$, we get $\mathcal{M}, x \models \psi \mathcal{P} \chi$.

The case for $\varphi = \psi \mathcal{R} \chi$ is immediate by Lemma 17 and the previous case. \blacktriangleleft

The main part left in this section is to show that filtrations exist. This is more complicated than in the purely quasi-discrete case, due to the different behaviour of topological paths. However, if we restrict ourselves to *finite* sets Σ , then we can normalise the paths on the

filtration according to Lemma 28, and use these simpler paths to establish satisfaction of the path modalities on the original model. Since we are only interested in filtrations through the set of subformulas induced by a single formula, this suffices for our purpose.

► **Lemma 32.** *Let Σ be a finite subformula closed set of formulas and \mathcal{M} a quasi-discrete model over topological paths. Furthermore, let X_Σ be the set of equivalence classes of \simeq_Σ , ν_Σ be defined as in Def. 30 (5), and $\eta_s([x]) = \langle \{[y] \mid \exists y', x': y' \in [y] \wedge x' \in [x] \wedge y \in N_{\min}(x)\} \rangle$ for each $[x] \in X_\Sigma$. Then the model $\mathcal{M}_\Sigma = ((X_\Sigma, \eta_s), \mathcal{I}_\mathbb{R}, \nu_\Sigma)$ is a filtration of \mathcal{M} over topological paths through Σ .*

Proof. First observe that \mathcal{M}_Σ is indeed a quasi-discrete neighbourhood model over topological paths, since the underlying space of \mathcal{M}_Σ is finite, and any finite neighbourhood space is quasi-discrete. We focus only on proving property 4 as all the others are already proved in Lemma 24.

Let $\pi: [0, 1] \rightarrow X_f$ be a path as required. If $n = 0$ so that $\mathcal{M}, x_n \models \psi$, this means $\mathcal{M}, x_0 \models \psi$, and so trivially $\mathcal{M}, x_0 \models \varphi \mathcal{P} \psi$. So, without loss of generality, we assume $n = 1$. With $\pi(i) = [x_i]$, we have $\mathcal{M}, x_i \models \varphi$ for $0 < i < 1$ and $\mathcal{M}, x_1 \models \psi$. Since the set of equivalence classes is finite, we can use Lemma 28 to get a path $\sigma: [0, 1] \rightarrow X_f$, with $x_0 \in \sigma(0)$ and $x_1 \in \sigma(1)$. Furthermore, the properties of σ in Lemma 28 ensure that for all $0 < i < 1$, if $\sigma(i) = [x'_i]$, then $\mathcal{M}, x'_i \models \varphi$.

Now, let $S = \{[z] \mid \exists i: \sigma(i) = [z]\}$ be the image of σ . Since S is finite, we define an order on S by setting $[z_i] < [z_j]$ iff there exist s and t with $s < t$ such that $\sigma(s) = [z_i]$ and $\sigma(t) = [z_j]$. By Lemma 28 and since the index space is totally ordered, this order is well-defined. So, in the following we will denote S by the sequence $[z_0], [z_1], \dots, [z_r]$.

We proceed to prove that $\mathcal{M}, x_0 \models \varphi \mathcal{P} \psi$ by induction on then length r of this sequence. If $r = 0$, then $[z_0] = [x_1]$. Since $z_0 \simeq x_0 \simeq x_1$ and $\mathcal{M}, x_1 \models \psi$, we get $\mathcal{M}, x_0 \models \psi$, and thus $\mathcal{M}, x_0 \models \varphi \mathcal{P} \psi$.

Assume that the property holds for all such sequences for a length up to r , and consider $[z_0], [z_1], [z_2], \dots, [z_r], [z_{r+1}]$. First, we can see that since σ is a path, the sequence $[z_1], [z_2], \dots, [z_r], [z_{r+1}]$ also induces a path that satisfies the precondition of the property. So, we get by the induction hypothesis $\mathcal{M}, z_1 \models \varphi \mathcal{P} \psi$. We now need to examine the relation between $[z_0]$ and $[z_1]$. To that end, we first consider the preimages of both classes: $I_0 = \{i \mid \sigma(i) = [z_0]\}$ and $I_1 = \{i \mid \sigma(i) = [z_1]\}$. Furthermore, let j be the supremum of I_0 . Recall that by Lemma 28, we have a sequence of indices i_0, i_1, \dots that partitions the interval $[0, 1]$ according to the values of σ . Now there are two possibilities for the relation between $[z_0]$ and $[z_1]$ according to σ .

1. If $i \in I_0$, then either $i = i_0 = 0$, or $i = i_1$. In the first case, $[z_0] = \sigma(i_0) \neq \sigma[i_0, i_1] = [z_1]$, and so $[z_1] \in N_{\min}([z_0])$ by Lemma 28 (5). In the other case, we have $[z_1] = \sigma[i_1, i_2]$, and so $[z_0] = \sigma(i_1) \neq \sigma[i_1, i_2] = [z_1]$. Again, by Lemma 28 (5), we have $[z_1] \in N_{\min}([z_0])$. By construction of \mathcal{M}_f there are $y_0, y_1 \in \mathcal{M}$ such that $y_1 \in N_{\min}(y_0)$ and $y_0 \in [z_0]$ and $y_1 \in [z_1]$. By assumption, we have $\mathcal{M}, x_0 \models \varphi$ as well, so by $x_0 \simeq z_0 \simeq y_0$, we get $\mathcal{M}, y_0 \models \varphi$ and $\mathcal{M}, y_1 \models \varphi \mathcal{P} \psi$. Then we have $\mathcal{M}, y_0 \models \varphi \mathcal{P} \psi$ from Lemma 29 (1) and thus $\mathcal{M}, x_0 \models \varphi \mathcal{P} \psi$.
2. Otherwise, we have $i \notin I_0$, and thus $i \in I_1$. Then certainly $i = i_1$, and so $[z_1] = \sigma(i_1) \neq \sigma[i_0, i_1] = [z_0]$. By Lemma 28 (4), we get $[z_0] \in N_{\min}([z_1])$. By construction of \mathcal{M}_f there are $y_0, y_1 \in \mathcal{M}$ such that $y_0 \in N_{\min}(y_1)$ and $y_0 \in [z_0]$ and $y_1 \in [z_1]$. However, in this case we also have that $i_1 > 0$, since otherwise $[z_0] = [z_1]$, which contradicts Property 3 of Lemma 28. So there is an $x \in [z_0]$, such that $\mathcal{M}, x \models \varphi$ by the properties of σ . Since $x \simeq y_0$, this means $\mathcal{M}, y_0 \models \varphi$. By assumption on σ , we have $\mathcal{M}, y_1 \models \varphi$ and since $y_1 \simeq z_1$, we also have $\mathcal{M}, y_1 \models \varphi \mathcal{P} \psi$. So, Lemma 29 (2) gives us $\mathcal{M}, y_0 \models \varphi \mathcal{P} \psi$, and with $x_0 \simeq z_0 \simeq y_0$ we can conclude the proof. ◀

The definition of filtrations together with Lemmas 31 and 32 yield the finite model property. Note that we can apply Lemma 32, as the set of subformulas of a formula is finite.

► **Theorem 33.** *If φ is a SLCS formula that is satisfiable on a quasi-discrete neighbourhood model over topological paths, then φ is satisfiable on a finite quasi-discrete neighbourhood model over topological paths.*

6 Conclusion

We have shown that SLCS does not have the finite model property over arbitrary neighbourhood models. Furthermore, we have proven that even when restricting to only quasi-discrete paths, there are still formulas that can only be satisfied on infinite models. Finally, we have shown that SLCS has the finite model property over models with underlying quasi-discrete neighbourhood spaces and quasi-discrete or topological paths. These results highlight that the types of spaces allowed have a much stronger impact on the existence of finite models than the types of paths allowed.

Our results are specific to the two types of paths we analysed. While these are the most common ones, it is possible to consider other definitions. Bubenik and Milićević [5] introduced other types of paths over neighbourhood spaces and analysed their properties. For example, they defined an index space based on a finite set $J = \{1, \dots, m\}$, which is close to the idea of a quasi-discrete space. However, the neighbourhood system on this index space is very different from our setting, since it includes both the predecessor and the successor in the minimal neighbourhood of a point. Several of their other index spaces are even more different. An interesting research direction for future work is to study how these types of paths interact with the operators of SLCS.

A more applied strand of research is to analyse some of the extensions of SLCS. A natural first step would be to consider the temporal extension of SLCS with operators from CTL [10] and prove whether it has the finite model property. This would build upon previous results stating that CTL has the finite model property [15] and the combinations of logics that admit finite models typically also admit finite models [13]. Similarly, interesting future work would be to analyse the extension of SLCS with set-based operators introduced by Ciancia et al. [11], and the metric extensions by Bartocci et al. [1]. Finally, a model-theoretic study of a variant of SLCS presented by Bezhanišvili et al. would be interesting [3]. This variant is defined with a semantics based on polyhedra in continuous spaces, which is in some sense “in between” the class of quasi-discrete, graph-like models, and the class of general, arbitrary neighbourhood spaces.

Our results are a further step towards a comprehensive model theory for SLCS. Understanding how the models of SLCS behave can guide how and where we may apply this logic, as well as its extensions.

References

- 1 Ezio Bartocci, Luca Bortolussi, Michele Loreti, and Laura Nenzi. Monitoring Mobile and Spatially Distributed Cyber-physical Systems. In *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*, MEMOCODE '17, pages 146–155, New York, NY, USA, 2017. ACM. event-place: Vienna, Austria. doi:10.1145/3127041.3127050.
- 2 Yuliy Baryshnikov and Robert Ghrist. Target enumeration via euler characteristic integrals. *SIAM J. Appl. Math.*, 70(3):825–844, 2009. doi:10.1137/070687293.

- 3 Nick Bezhanishvili, Vincenzo Ciancia, David Gabelaia, Gianluca Grilletti, Diego Latella, and Mieke Massink. Geometric model checking of continuous space. *CoRR*, abs/2105.06194, 2021. [arXiv:2105.06194](https://arxiv.org/abs/2105.06194).
- 4 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001. [doi:10.1017/CB09781107050884](https://doi.org/10.1017/CB09781107050884).
- 5 Peter Bubenik and Nikola Milićević. Applied topology using Čech’s closure spaces, 2021. [arXiv:2104.10206](https://arxiv.org/abs/2104.10206).
- 6 Fabrizio Banci Buonamici, Gina Belmonte, Vincenzo Ciancia, Diego Latella, and Mieke Massink. Spatial logics and model checking for medical imaging. *Int. J. Softw. Tools Technol. Transf.*, 22(2):195–217, 2020. [doi:10.1007/s10009-019-00511-9](https://doi.org/10.1007/s10009-019-00511-9).
- 7 Davide Castelnovo and Marino Miculan. Closure hyperdoctrines, with paths. *CoRR*, abs/2007.04213, 2020. [arXiv:2007.04213](https://arxiv.org/abs/2007.04213).
- 8 Gustave Choquet. Convergences. *Annales de l’université de Grenoble. Nouvelle série. Section sciences mathématiques et physiques*, 23:57–112, 1947-1948. URL: http://www.numdam.org/item/AUG_1947-1948__23__57_0/.
- 9 Vincenzo Ciancia, Stephen Gilmore, Gianluca Grilletti, Diego Latella, Michele Loreti, and Mieke Massink. Spatio-temporal model checking of vehicular movement in public transport systems. *International Journal on Software Tools for Technology Transfer*, 20:289–311, January 2018. [doi:10.1007/s10009-018-0483-8](https://doi.org/10.1007/s10009-018-0483-8).
- 10 Vincenzo Ciancia, Gianluca Grilletti, Diego Latella, Michele Loreti, and Mieke Massink. An experimental spatio-temporal model checker. In *Software Engineering and Formal Methods. SEFM 2015*, volume 9509 of *Lecture Notes in Computer Science*. Springer, 2015. [doi:10.1007/978-3-662-49224-6_24](https://doi.org/10.1007/978-3-662-49224-6_24).
- 11 Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. Model Checking Spatial Logics for Closure Spaces. *Logical Methods in Computer Science*, Volume 12, Issue 4, 2017. [doi:10.2168/LMCS-12\(4:2\)2016](https://doi.org/10.2168/LMCS-12(4:2)2016).
- 12 Vincenzo Ciancia, Diego Latella, Mieke Massink, and Erik de Vink. Towards spatial bisimilarity for closure models: Logical and coalgebraic characterisations, 2020. [arXiv:2005.05578](https://arxiv.org/abs/2005.05578).
- 13 M. E. Coniglio, A. Sernadas, and C. Sernadas. Preservation by fibring of the finite model property. *Journal of Logic and Computation*, 21(2):375–402, 2011. [doi:10.1093/logcom/exq022](https://doi.org/10.1093/logcom/exq022).
- 14 Szymon Dolecki and Frédéric Mynard. *Convergence Foundations of Topology*. World Scientific, 2016. [doi:10.1142/9012](https://doi.org/10.1142/9012).
- 15 E. Allen Emerson and Jai Srinivasan. Branching time temporal logic. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 123–172, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.
- 16 D. C. Kent and Won Keun Min. Neighborhood spaces. *International Journal of Mathematics and Mathematical Sciences*, 32(7):387–399, 2002. [doi:10.1155/S0161171202202203](https://doi.org/10.1155/S0161171202202203).
- 17 Christine Largeron and Stéphane Bonnevey. A pretopological approach for structural analysis. *Information Sciences*, 144(1-4):169–185, 2002.
- 18 Sven Linker, Fabio Papacchini, and Michele Sevegnani. Analysing spatial properties on neighbourhood spaces. In Javier Esparza and Daniel Král’, editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPICs*, pages 66:1–66:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. [doi:10.4230/LIPICs.MFCS.2020.66](https://doi.org/10.4230/LIPICs.MFCS.2020.66).
- 19 Sven Linker and Michele Sevegnani. Target counting with presburger constraints and its application in sensor networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2231):20190278, 2019. [doi:10.1098/rspa.2019.0278](https://doi.org/10.1098/rspa.2019.0278).
- 20 Antonio Rieser. Čech closure spaces: A unified framework for discrete and continuous homotopy. *Topology and its Applications*, 296:107613, 2021. [doi:10.1016/j.topol.2021.107613](https://doi.org/10.1016/j.topol.2021.107613).

72:16 Finite Models for a Spatial Logic with Discrete and Topological Path Operators

- 21 Lynn Arthur Steen and J. Arthur Seebach, Jr. *Counterexamples in Topology*. Springer-Verlag, New York, 1978. Reprinted by Dover Publications, New York, 1995.
- 22 Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. Modeling and verification of evolving cyber-physical spaces. In Matthias Tichy, Eric Bodden, Marco Kuhrmann, Stefan Wagner, and Jan-Philipp Steghöfer, editors, *Software Engineering und Software Management 2018, Fachtagung des GI-Fachbereichs Softwaretechnik, SE 2018, 5.-9. März 2018, Ulm, Germany*, volume P-279 of *LNI*, pages 113–114. Gesellschaft für Informatik, 2018. URL: <https://dl.gi.de/20.500.12116/16369>.
- 23 Eduard Čech. *Topological spaces*. Academia, Publishing House of the Czechoslovak Academy of Sciences, 1966. Edited by Zdeněk Frolík and Miroslav Katětov.

Recursive Backdoors for SAT

Nikolas Mählmann  

University of Bremen, Germany

Sebastian Siebertz  

University of Bremen, Germany

Alexandre Vigny  

University of Bremen, Germany

Abstract

A strong backdoor in a formula φ of propositional logic to a tractable class \mathcal{C} of formulas is a set B of variables of φ such that every assignment of the variables in B results in a formula from \mathcal{C} . Strong backdoors of small size or with a good structure, e.g. with small backdoor treewidth, lead to efficient solutions for the propositional satisfiability problem SAT.

In this paper we propose the new notion of *recursive backdoors*, which is inspired by the observation that in order to solve SAT we can independently recurse into the components that are created by partial assignments of variables. The quality of a recursive backdoor is measured by its *recursive backdoor depth*. Similar to the concept of backdoor treewidth, recursive backdoors of bounded depth include backdoors of unbounded size that have a certain treelike structure. However, the two concepts are incomparable and our results yield new tractability results for SAT.

2012 ACM Subject Classification Theory of computation \rightarrow Logic; Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Propositional satisfiability SAT, Backdoors, Parameterized Algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.73

Related Version *Previous Version:* <https://arxiv.org/abs/2102.04707>

Funding This research is funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) – Projektnummer 444419611.

1 Introduction

The problem of checking whether a formula of propositional logic in conjunctive normal form (CNF) is satisfiable (SAT) is one of the most central problems in computer science. The problem is often seen as the canonical NP-complete problem [1] and conjectured to be not solvable in sub-exponential time [10]. Despite this theoretical hardness result, state-of-the-art SAT solvers are able to efficiently solve multi-million variable instances arising from real-world applications. We refer to the recent survey of Ganesh and Vardi [4], who try to explain this “unreasonable effectiveness of SAT solvers”. SAT is known to be solvable in polynomial time on several restricted classes of formulas, e.g. on Horn and 2CNF formulas. However, this classification falls short of explaining the practical efficiency of SAT solvers, as many efficiently solvable instances do not belong to any of these classes.

Parameterized complexity theory offers a refined view on the complexity of problems. Instead of measuring complexity only with respect to the input size n , one or more parameters are taken into account. Optimally, one can establish fixed-parameter tractability with respect to a parameter k , that is, a running time of $f(k) \cdot n^c$ for some computable function f and a constant c . In case the parameter k is small on a given class of instances, this may lead to efficient algorithms even if the inputs are large. Even though SAT solvers may not be explicitly tailored to use these parameters, it is conceivable that they implicitly exploit the



© Nikolas Mählmann, Sebastian Siebertz, and Alexandre Vigny;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 73; pp. 73:1–73:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

structure that is imposed by them. This poses the question of parametric characterizations of real-world application instances that can be solved efficiently. One very important parameter to explain tractability is *treewidth*, which intuitively measures how tree-like an instance is, and which can be used to obtain fixed-parameter tractability for SAT [15].

A second very successful parametric approach was introduced by Williams et al. [17]. For a formula φ , a *strong backdoor* to a given class \mathcal{C} of formulas is a set of variables of φ such that for every assignment of these variables one obtains a formula in \mathcal{C} . Similarly, a *weak backdoor* to \mathcal{C} for a satisfiable formula is a set of variables of φ such that some assignment of these variables leads to a formula in \mathcal{C} . These notions elegantly allow to lift tractability results from classes \mathcal{C} to classes that are *close* to \mathcal{C} . Given a formula and a strong backdoor of size k to a tractable class, one can decide satisfiability by checking 2^k tractable instances. For small k this yields efficient algorithms as noted by Nishimura et al. [11], who first studied the parameterized complexity of backdoor detection.

A lot of effort has been invested to develop fpt algorithms for backdoor detection to various tractable base classes \mathcal{C} , for example to classes of bounded treewidth [9] or heterogeneous classes [7]. Treewidth is a width measure for graphs that can however be applied to measure the complexity of formulas by considering the incidence graphs of formulas. The incidence graph of a formula has one vertex for each variable and one vertex for each clause. A variable vertex is connected with a clause vertex when the variable is contained positively or negatively in the clause. In the following, we will often use graph theoretic terminology for formulas, and this always refers to the incidence graph of the formula.

Apart from various base classes, alternative measures of quality of backdoors have been proposed. Backdoor trees generalize backdoor sets into decision trees, whose quality is measured by their number of leaves [14]. Recently backdoor trees have been further generalized to backdoor DNFs [12]. Ganian et al. [5] introduced the notion of backdoor treewidth, which permits fpt backdoor detection for backdoors of unbounded size. Even though backdoors of bounded treewidth can be arbitrarily large, they showed that SAT is fixed-parameter tractable when parameterized by the backdoor treewidth with respect to the classes \mathcal{C} of Horn, Anti-Horn and 2CNF formulas. They also consider backdoors that split an input constraint satisfaction problem into components that may belong to different tractable classes [6]. Other recent notions of backdoors can be found in the literature such as the notions of learning-sensitive backdoors [3] and learning-sensitive backdoors with restarts [18]. For an overview of additional works we refer to the survey by Gaspers and Szeider [8] as well as to the upcoming book chapter by Samer and Szeider [16].

In this paper we introduce the new notions of *strong* and *weak recursive backdoors* as generalizations of backdoor sets and backdoor trees. Strong recursive backdoors extend backdoor trees by not only branching on truth values but also recursively branching into the independent components of the formula that may arise after the partial assignment of variables. We measure the quality of recursive backdoors by the depth of their branching trees. The splitting into components allows recursive backdoors of bounded depth to contain an unbounded number of variables. Our definition, together with the observation that after the assignment of a variable one can independently solve the sub-instances in the arising components, reveals a new potential of backdoors for SAT.

The main power of recursive backdoors, but also the difficulty in their study, is that by assigning a variable we do not recurse into the components that are created by deleting that variable, but into the components that are created by deleting parts of the neighborhood of the variable. We show that detecting weak recursive backdoors even to the class \mathcal{C}_0 of

edgeless incidence graphs is $W[2]$ -hard. Our main technical contribution is an fpt algorithm that, given a formula φ and a parameter k , either decides satisfiability of φ or correctly concludes that φ has no strong recursive backdoor to \mathcal{C}_0 of depth at most k . Even for the class \mathcal{C}_0 this yields tractability results that cannot be achieved by backdoor treewidth.

We provide background in Section 2. We define recursive backdoors in Section 3 and Section 4 is devoted to a sketch of the fpt algorithm. The rest of the paper is devoted to the formal presentation and correctness proof of that algorithm. Due to space constraints we present the hardness proof only in the appended full version of the paper. Also some proofs of the main result are deferred to the appendix.

2 Preliminaries

Propositional Logic. We consider formulas of propositional logic in conjunctive normal form (CNF), represented by finite sets of clauses, and in the following when we speak of a formula we will always mean a CNF formula. We write $x, y, z \dots$ for variables and $\star, \diamond \in \{+, -\}$ for polarities. A literal is a variable with an assigned polarity. We write x_+ for the positive literal x , x_- for the negative literal \bar{x} , and x_\star for a literal with arbitrary polarity. Every clause is a finite set of literals. We assume that no clause contains a complementary pair x_+, x_- . For a formula φ , we write $\text{var}(\varphi)$ and $\text{cl}(\varphi)$ to refer to the sets of variables and clauses of φ , respectively. We say that a variable x is positive (resp. negative) in a clause c if $x_+ \in c$ (resp. $x_- \in c$), and we write $\text{var}(c)$ for the set of variables x with $x_\star \in c$ and $\text{lit}(c)$ for the set of literals in c . For a formula φ we let $\text{var}(\varphi) = \bigcup_{c \in \varphi} \text{var}(c)$.

The *width* of c is $|\text{var}(c)|$ and the *length* of φ is $\sum_{c \in \varphi} |\text{var}(c)|$, denoted $|c|$ and $|\varphi|$, respectively. We call a clause a *d-clause* if it has width exactly d . We say that a formula has *maximal clause degree* d if each of its clauses has width at most d . We write \mathcal{C}_d to refer to the class of CNF formulas with maximal clause degree d . Especially \mathcal{C}_0 denotes the class of empty formulas, that either contain only empty clauses or no clauses at all.

A *truth assignment* τ is a mapping from a set of variables, denoted by $\text{var}(\tau)$, to $\{+, -\}$. A truth assignment τ satisfies a clause c if c contains at least one literal x_\star with $\tau(x) = \star$. A truth assignment τ of $\text{var}(\varphi)$ satisfies the formula φ if it satisfies all clauses of φ .

Given a formula φ and a truth assignment τ , $\varphi[\tau]$ denotes the formula obtained from φ by removing all clauses that are satisfied by τ and by removing from the remaining clauses all literals x_\star with $\tau(x) \neq \star$. Note that for every formula φ and assignment τ we have $\text{var}(\varphi[\tau]) \cap \text{var}(\tau) = \emptyset$. If τ and τ' are assignments with $\text{var}(\tau) \cap \text{var}(\tau') = \emptyset$, then we write $\tau \cup \tau'$ for the unique assignment extending both τ and τ' .

Graphs. We will only consider graphs that arise as incidence graphs of formulas. The incidence graph G_φ of a formula φ is a bipartite graph with vertices $\text{var}(\varphi) \cup \text{cl}(\varphi)$. Slightly abusing notation we usually do not distinguish between a formula and its incidence graph. E.g. we speak of the variables and clauses of G_φ , which we denote by $\text{var}(G_\varphi)$ and $\text{cl}(G_\varphi)$ respectively. Vice versa, we speak e.g. of components of φ with implicit reference to the incidence graph G_φ . We drop the subscript φ if it is clear from the context. The edges of G are partitioned into two parts E_+ (positive edges) and E_- (negative edges), where a variable x is connected to a clause c by an edge E_\star if $x_\star \in \text{lit}(c)$. For an assignment τ we naturally define $G[\tau]$ as the incidence graph of $\varphi[\tau]$. If τ assigns only a single variable $x \mapsto \star$ we write $G[x_\star]$ for $G[\tau]$. Note that for every assignment τ , $G[\tau]$ is an induced subgraph of G . For a vertex v the closed \star -neighborhood of v is defined as $N_\star[v] := \{w : \{v, w\} \in E_\star\}$. For $W \subseteq V$ we write $G[W]$ for the subgraph induced by W and $G - W$ for $G[V \setminus W]$.

We refrain from formally defining treewidth and refer to the literature for background. A graph H is a minor of a graph G if H can be obtained from G by deleting edges and vertices and by contracting edges. To compare our new definition of recursive backdoors with backdoor treewidth, we mention that if a graph contains a $k \times k$ grid as a minor, then it has treewidth at least k [13].

Parameterized Complexity. A parameterized problem is called fixed-parameter tractable (fpt) if there exists an algorithm deciding the problem in time $f(k) \cdot n^c$, where n is the input size, k is the parameter, f is a computable function and c is a constant. An algorithm witnessing fixed-parameter tractability of a problem is called an fpt-algorithm for the problem.

To show that a problem is likely to not be fpt one can show that it is $W[i]$ -hard for some $i \geq 1$. For this, it is sufficient to give a parameterized reduction from a known $W[i]$ -hard problem. We refer to the book [2] for extensive background on parameterized complexity theory.

Backdoors. Let \mathcal{C} be a class of formulas and let φ be a formula. A set $B \subseteq \text{var}(\varphi)$ is a *strong backdoor* of φ to \mathcal{C} if for every assignment $\tau: B \rightarrow \{+, -\}$ the formula $\varphi[\tau]$ belongs to \mathcal{C} . Note that for some assignments τ the formula $\varphi[\tau]$ may not be satisfiable, even though φ is satisfiable. Hence, in the following definition of a weak backdoor we require that φ is satisfiable. If φ is satisfiable, then a set $B \subseteq \text{var}(\varphi)$ is a *weak backdoor* of φ to the class \mathcal{C} if there exists an assignment $\tau: B \rightarrow \{+, -\}$ such that $\varphi[\tau]$ is a satisfiable formula in \mathcal{C} . The classical measure for the complexity or quality of a backdoor is its size.

An important recent approach to measure the complexity of a backdoor is to take its structure into account. The *treewidth of a backdoor* B is defined as the treewidth of the graph with vertex set B where two variables x and y are connected by an edge if there exists a path from a neighbor of x to a neighbor of y in $G - B$ [5]. Ganian et al. [5] also consider backdoors that split the input CNF formula into components that each may belong to a different tractable class \mathcal{C} .

Permissive Backdoor Detection. In their survey Gaspers and Szeider [8] differ between a strict and a permissive version of the backdoor detection problem. Given a backdoor definition \mathcal{B} and a corresponding quality measure μ (e.g. strong backdoors to 2CNF measured by their size) as well as a formula φ and a parameter k . The strict backdoor detection problem, denoted as \mathcal{B} -DETECTION, asks whether or not $\mu(\varphi) \leq k$ holds. The permissive backdoor detection problem, denoted as $\text{SAT}(\mu)$, asks to either decide the satisfiability of φ or conclude that $\mu(\varphi) > k$ holds. The permissive version of the problem grants more freedom in algorithm design, as trivial instances can be solved without calculating the backdoor measure. However as Gaspers and Szeider point out, hardness proofs seem to be much harder for the permissive version.

3 Recursive Backdoors

Strong Recursive Backdoors. Our new concept of recursive backdoors is based on the observation that we can handle the components of $G[x_\star]$ independently whenever a variable x has been assigned. A *strong recursive backdoor* (SRB) of an incidence graph G to a class \mathcal{C} is a rooted labeled tree, where every node is either labeled with a subgraph of G or with a variable in $\text{var}(G)$. The root of the tree is labeled with G . Whenever an inner node is labeled with a connected graph H , then it has one child labeled with a variable. Whenever

it is labeled with a disconnected graph, then it has one child for each of its components, labeled with the graph induced by that component. Whenever an inner node is labeled with a variable x , then its parent is labeled with a graph H , and its two children are labeled with $H[x_+]$ and $H[x_-]$, respectively. Every leaf node is labeled with a graph from \mathcal{C} . We call the nodes of the tree *variable nodes* or *component nodes*, according to their labeling.

The *depth* of a strong recursive backdoor is the maximal number of variable nodes from its root to one of its leaves. The *strong recursive backdoor depth* to a class \mathcal{C} ($\text{srbd}_{\mathcal{C}}$) of an incidence graph G is the minimal depth of a strong recursive backdoor of G to \mathcal{C} . We give the following equivalent definition:

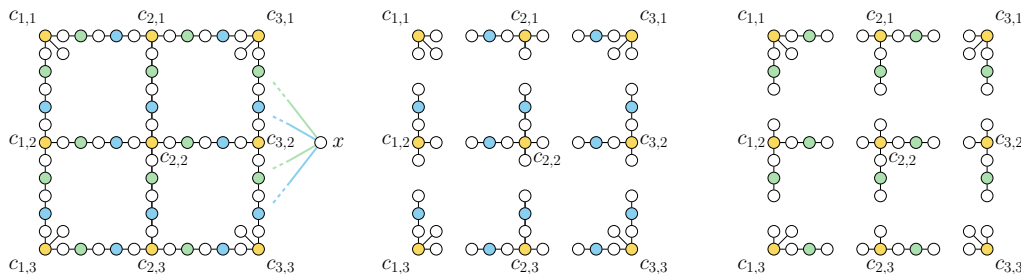
► **Definition 3.1** (Strong Recursive Backdoor Depth).

$$\text{srbd}_{\mathcal{C}}(G) = \begin{cases} 0 & \text{if } G \in \mathcal{C} \\ 1 + \min_{x \in \text{var}(G)} \max_{\star \in \{+, -\}} \text{srbd}_{\mathcal{C}}(G[x_{\star}]) & \text{if } G \notin \mathcal{C} \text{ and } G \\ & \text{is connected} \\ \max \{ \text{srbd}_{\mathcal{C}}(H) : H \text{ connected component of } G \} & \text{otherwise} \end{cases}$$

To get a better understanding of strong recursive backdoor depth we give an example of a family of incidence graphs with unbounded backdoor treewidth to 2CNF but constant strong recursive backdoor depth to \mathcal{C}_0 , the class of edgeless graphs. For any $k \geq 0$, define the graph G_k as follows. We start with a $k \times k$ grid of clause vertices $\{c_{1,1}, \dots, c_{k,k}\}$, depicted in yellow in Figure 1. We connect a private variable vertex to each of the corners $c_{1,1}, c_{1,k}, c_{k,1}, c_{k,k}$ of the grid. We now replace each edge of the grid by a path of length 6 (containing 5 vertices). Every second vertex on a new path is a clause vertex, connected to the two adjacent variable vertices. Furthermore, we add a special variable vertex x that is connected alternatingly with positive and negative polarity (depicted in green and blue in the figure) to the clause vertices on the new paths. Variable vertices are depicted as white vertices in the figure.

Since every clause $c_{i,j}$ is connected to at least 3 variables, every backdoor set B to 2CNF will have to contain at least one variable of every $c_{i,j}$. This implies that the B -torso of G will always contain a $k \times k$ grid as a minor, hence, will have treewidth at least k . We refer to [5, Definition 1] for the precise notions of B -torso and *backdoor treewidth*.

A strong recursive backdoor to \mathcal{C}_0 with x as its root splits every grid clause into a separate component of constant size and therefore has constant depth.



■ **Figure 1** From left to right: G_3 , $G_3[x_+]$, and $G_3[x_-]$.

Conversely, for the base class \mathcal{C}_0 , formulas whose incidence graph is a long path, e.g. $(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee \dots \vee (x_n \wedge x_{n+1})$, have constant backdoor treewidth, but unbounded strong recursive backdoor depth, as we will see in Lemma 5.2. We conclude that recursive backdoors and backdoor treewidth are incomparable.

SAT Solving and SAT Counting using Strong Recursive Backdoors. Similar to regular strong backdoor sets, strong recursive backdoors allow for polynomial time SAT Solving and SAT Counting if the backdoor is given as part of the input. First, we observe that even though it may have an unbounded branching degree, the size of a recursive backdoor is still linear in the size of its formula.

► **Lemma 3.2.** *Let T be a strong recursive backdoor with depth k of a formula φ to a class \mathcal{C} . The number of leaf nodes in T , as well as the sum of number of vertices contained in leaf nodes is bound by $2^k \cdot |\varphi|$.*

Proof. Proof by induction on k and $|\varphi|$. The bound trivially holds when $k = 0$ or $|\varphi| = 1$ and the backdoor consists of a single leaf node.

In the inductive step, a variable node increases the backdoor depth and at most doubles the number of leaves and their contained vertices, as it branches on both polarities. A component node does not increase the backdoor depth, but branches over disjoint components of strictly smaller size. As the sum of the vertices contained in the components is equal to $|\varphi|$, the number of leaves and contained vertices is again bounded by $2^k \cdot |\varphi|$. ◀

We can use this observation to construct a straight-forward bottom-up algorithm:

► **Proposition 3.3.** *For every class \mathcal{C} where satisfiability checking (resp. counting the number of satisfying assignments) can be done in polynomial time, for every formula φ and integer k , given a strong recursive backdoor with depth k of φ to \mathcal{C} , we can test the satisfiability (resp. count the number of satisfying assignments) of φ in time $2^k \cdot \text{poly}(|\varphi|)$.*

Proof. By Lemma 3.2, we know that we have at most $2^k \cdot |\varphi|$ instances labeling leaves, which can be solved in polynomial time. For variable nodes, the instance labeling the node is satisfiable if and only if at least one of its two children is labeled with a satisfiable instance. The number of satisfiable assignments is the sum of the satisfiable assignments for its children. For component nodes, the instance labeling the node is satisfiable if and only if all of its children are labeled with satisfiable instances. The number of satisfiable assignments is the product of the satisfiable assignments for its children. ◀

Weak Recursive Backdoors. Recall that in the definition of weak backdoors we consider only satisfiable formulas and aim to find an assignment τ that leads to a satisfiable formula $\varphi[\tau] \in \mathcal{C}$. This is also the case in the following definition of weak recursive backdoor depth:

► **Definition 3.4** (Weak Recursive Backdoor Depth).

$$\text{wrbd}_{\mathcal{C}}(G) = \begin{cases} 0 & \text{if } G \in \mathcal{C} \text{ and } G \text{ is satisfiable} \\ \infty & \text{if } G \in \mathcal{C} \text{ and } G \text{ is unsatisfiable} \\ 1 + \min_{x \in \text{var}(G)} \min_{\star \in \{+, -\}} \text{wrbd}_{\mathcal{C}}(G[x_{\star}]) & \text{if } G \notin \mathcal{C} \text{ and } G \text{ is connected} \\ \max \{ \text{wrbd}_{\mathcal{C}}(H) : H \text{ connected component of } G \} & \text{otherwise} \end{cases}$$

SAT Solving using Weak Recursive Backdoors. We can use the notion of weak recursive backdoors for SAT as follows. As the existence of a weak recursive backdoor for some formula φ implies that φ is satisfiable, we do not assume that the backdoor is given with the input this time.

► **Proposition 3.5.** *For every class \mathcal{C} for which we can test membership and satisfiability in polynomial time, for every formula φ and integer k , we can test whether φ has weak recursive backdoor depth at most k in time $(2 \cdot |\varphi|)^k \cdot \text{poly}(|\varphi|)$.*

Proof. Branch over all $2 \cdot |\varphi|$ possible truth assignments of a single variable in φ . Recurse into the components arising through the assignment, until branching depth k or a formula from \mathcal{C} is reached. The leaves of the branching tree are labeled as satisfiable, if they are satisfiable members of \mathcal{C} , which can be tested in time $\text{poly}(|\varphi|)$. Component (resp. variable) branching nodes are labeled as satisfiable, if all (resp. any) of their children are labeled as satisfiable. It is now easy to see that $\text{wrbd}_{\mathcal{C}}(\varphi) \leq k$ if and only if the root node of the branching tree is labeled as satisfiable. By the same argument as in Lemma 3.2, the branching tree has at most $(2 \cdot |\varphi|)^k \cdot |\varphi|$ leaves. Therefore the presented algorithm runs in time $(2 \cdot |\varphi|)^k \cdot \text{poly}(|\varphi|)$. ◀

Note that when k is small, even this running time is a major improvement over the worst case running time of 2^{cn} implied by the exponential time hypothesis (ETH).

We will now continue with a sketch of the fpt algorithm for strong recursive backdoor detection to the class of empty formulas.

4 Proof Sketch

Our goal is to show that the permissive backdoor detection problem $\text{SAT}(\text{srbd}_{\mathcal{C}_0})$ is fixed-parameter tractable. That is, we aim to decide for a given formula φ and parameter k whether φ is satisfiable or does not have a strong recursive backdoor of depth k to the class \mathcal{C}_0 of empty formulas i.e. $\text{srbd}_{\mathcal{C}_0}(\varphi) > k$. Our approach is based on two main observations: Formulas with strong recursive backdoor depth k to \mathcal{C}_0 have both a maximal clause degree k (see Lemma 5.1) and a diameter bounded by $\lambda_k := 4 \cdot 2^k$ in each connected component (see Lemma 5.2).

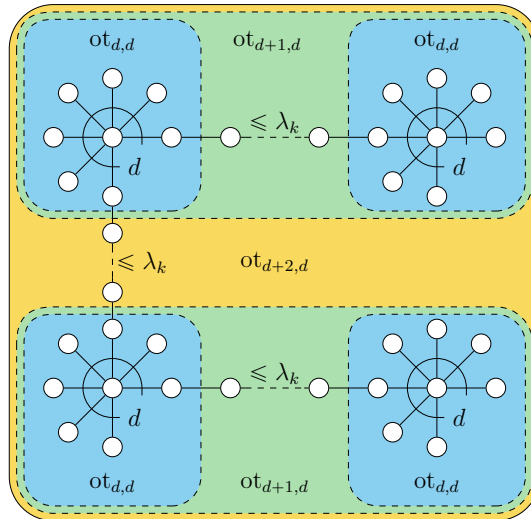
We are going to design a recursive algorithm that in every step finds a bounded depth SRB that reduces the maximal clause degree of φ by one, or proves that the strong recursive backdoor depth of φ is larger than k . We extend the SRB by recursively branching on its leaves until we reach \mathcal{C}_0 . Since φ has maximal clause degree k this yields a bounded depth SRB to \mathcal{C}_0 in fpt running time.

First, take a look at the special case where $G := G_\varphi$ contains a clause c of width k , i.e. a k -clause. By our first observation, the existence of c implies that $\text{srbd}_{\mathcal{C}_0}(G) \geq k$. Note that the degree of c can only be reduced by assigning a variable in its neighborhood.

Next, consider the case where G contains two disjoint $(k-1)$ -clauses c_1, c_2 in the same connected component. Since G has limited diameter, there exists a path P of length $\leq \lambda_k$ between them. Since c_1 and c_2 are part of the same component we are only allowed to assign one variable to reduce the backdoor depth of that component to $k-1$. No matter which variable we choose, since c_1 and c_2 are disjoint, one of them will continue to exist in the reduced graph, which will then have backdoor depth at least $k-1$ and again, the existence of c_1, c_2 , and P implies that $\text{srbd}_{\mathcal{C}_0}(G) \geq k$.

Given a maximal clause degree d , we generalize this strategy for arbitrary depths $k = d + j$ by searching for so called *obstruction-trees*. An obstruction-tree for depth k is a structured set of vertices, whose existence in G will guarantee that G has a strong recursive backdoor depth of at least k . We start by searching for d -clauses as obstruction-trees for depth d . We search for obstruction-trees for depth $d + j + 1$ by searching for two obstruction-trees of depth $d + j$ that have disjoint neighborhoods and are connected by a path of bounded length. A schematic depiction of obstruction-trees for maximal clause degree d and backdoor depths $d, d + 1$, and $d + 2$ is shown in Figure 2. Since the obstruction-trees are based on d -clauses we can construct an fpt algorithm that either finds an obstruction-tree or a small backdoor which reduces the maximal clause degree of G to $d - 1$.

The algorithm to find obstruction-trees, described in Proposition 5.9, is at the heart of our proof. We use this algorithm to solve G by recursively searching for obstruction-trees for depth $k + 1$. In each round we either abort and conclude that G has strong recursive backdoor depth at least $k + 1$ or reduce d and recurse until we arrive at \mathcal{C}_0 , where we can trivially check satisfiability. If the graph splits into multiple components we can handle the components separately and aggregate the results.



■ **Figure 2** Schematic depiction of an obstruction-tree for maximal clause degree d and strong recursive backdoor depth $d + 2$. Here, the notation “ $ot_{i,d}$ ” stands for the notion of (i, d, k) -obstruction-tree, as formally defined in Definition 5.3.

5 Permissive Strong Recursive Backdoor Detection to \mathcal{C}_0 Is FPT

We start by formalizing and proving the observations made in Section 4.

► **Lemma 5.1** (Limited Clause Degree). *For every incidence graph G and integer d , if $\text{srb}_{\mathcal{C}_0}(G) \leq d$, then G has maximal clause degree at most d , i.e. $G \in \mathcal{C}_d$.*

Proof. Proof by induction on d .

Base Case: $d = 0$. If $\text{srb}_{\mathcal{C}_0}(G) \leq 0$, then $G \in \mathcal{C}_0$.

Induction Step: Let d be an integer and G an incidence graph with $\text{srb}_{\mathcal{C}_0}(G) \leq d + 1$. Let c be any clause in G . Let H be the connected component of c in G . By assumption, $\text{srb}_{\mathcal{C}_0}(H) \leq d + 1$, and there must be a variable vertex x such that for every literal x_\star we have $\text{srb}_{\mathcal{C}_0}(H[x_\star]) \leq d$. By induction, we also have that $H[x_\star] \in \mathcal{C}_d$.

Case 1: x is not connected to c . Then c is still intact in $H[x_\star]$ and by induction contains at most d variables.

Case 2: x is connected to c by an edge with polarity $+$. Then c still exists in $H[x_-]$ and by induction has degree at most d in $H[x_-]$. Therefore c contains at most $d + 1$ variables in G .

Case 3: The case that x is connected to c by an edge with polarity $-$ is analogous to *Case 2*. In all cases, c contains at most $d + 1$ variables in G , and therefore $G \in \mathcal{C}_{d+1}$. ◀

► **Lemma 5.2** (Low Diameter). *Let G be a incidence graph. If either $\text{srbd}_{\varphi_0}(G) \leq k$ or $\text{wrbd}_{\varphi_0}(G) \leq k$, then every connected component of G has a diameter of at most $4 \cdot 2^k - 4$.*

Proof. Proof by induction on k . For brevity we write $\text{bd}(G) \leq k$ for the fact that either $\text{wrbd}_{\varphi_0}(G) \leq k$ or $\text{srbd}_{\varphi_0}(G) \leq k$.

Base Case: $k = 0$. In this case, G is edgeless, and the statement holds.

Induction Step: Assume towards a contradiction that $\text{bd}(G) \leq k + 1$ and that G has a connected component H of diameter at least $4 \cdot 2^{k+1} - 3$. Hence H contains two vertices connected by a shortest path $P = (v_1, \dots, v_m)$ with $m = 4 \cdot 2^{k+1} - 2$ (such that if v_i is a clause vertex, then v_{i+1} is a variable vertex, and if v_i is a variable vertex, then v_{i+1} is a clause vertex). Also there exists a literal y_\star such that y is a variable vertex in H witnessing that $\text{bd}(G) \leq k + 1$. Since P is a shortest path from v_1 to v_m , y can only be connected to at most 2 clauses in P at distance 2 from each other. Let v_{i-1} and v_{i+1} be the two clauses connected to y (the reasoning also works with only one or zero such v_j). Now assigning y_\star can split P by deleting v_{i-1} and v_{i+1} . We then have that $G[y_\star]$ still contains (v_1, \dots, v_{i-2}) and (v_{i+2}, \dots, v_m) as shortest paths. One of those two paths will include at least

$$\left\lceil \frac{m-3}{2} \right\rceil = \left\lceil \frac{4 \cdot 2^{k+1} - 2 - 3}{2} \right\rceil = \left\lceil 4 \cdot 2^k - \frac{5}{2} \right\rceil = 4 \cdot 2^k - 2$$

vertices. One component of $H[y_\star]$ therefore has a diameter of at least $4 \cdot 2^k - 3$. This contradicts the fact that, by induction, $\text{bd}(H[y_\star]) \leq k$. Therefore we have $\text{bd}(G) \leq k + 1$. ◀

In the rest of the paper, we use $\lambda_k = 4 \cdot 2^k$ for brevity.

5.1 Obstruction-Trees

We now turn to the concept of obstruction-trees. We first define them and then prove some of their properties. The main property, proved in Proposition 5.6, is that the existence of such trees witnesses a lower bound for the depth of a strong recursive backdoor to the class \mathcal{C}_0 .

► **Definition 5.3** (Obstruction-Trees and Destroy Neighborhoods). *For all integers k, d and incidence graphs G in \mathcal{C}_d , we inductively for $i \geq d$ define the notion of an (i, d, k) -obstruction-tree T of G with elements $V(T)$ and destroy-neighborhood $N_G^\dagger[T]$. We use $\text{cla}(T)$ and $\text{var}(T)$ to denote the clauses and variables of $V(T)$.*

For a set $T = \{c, x_1, \dots, x_d\}$, where c is a d -clause of G with $\text{var}(c) = \{x_1, \dots, x_d\}$, we have:

1. T is a (d, d, k) -obstruction-tree.
2. $V(T)$ are the elements of T .
3. $N_G^\dagger[T] := \text{var}(T) = \{x_1, \dots, x_d\}$.

Inductively, for a triple $T = (T_1, P, T_2)$ where T_1 and T_2 are (i, d, k) -obstruction-trees of G such that $N_G^\dagger[T_1] \cap N_G^\dagger[T_2] = \emptyset$, and P is a path of length at most λ_k connecting T_1 and T_2 , we have:

73:10 Recursive Backdoors for SAT

1. T is an $(i + 1, d, k)$ -obstruction-tree.
2. $V(T) := V(T_1) \cup V(P) \cup V(T_2)$.
3. $N_G^\dagger[T] := \text{var}(T) \cup \{x : \text{there exist } c_1, c_2 \in \text{cla}(T) \text{ with } \{x, c_1\} \in E_+ \text{ and } \{x, c_2\} \in E_-\}$.

Observe that $V(T)$ is a connected subset of G . We now show, that our definition of a destroy neighborhood is a small set of variables, that shields the obstruction-tree from the rest of the graph. Remember that $\lambda_k = 4 \cdot 2^k$.

► **Proposition 5.4** (N^\dagger Is Small). *For all integers i, d, k with $d \leq k$, for every incidence graph G in \mathcal{C}_d , and every (i, d, k) -obstruction-tree T of G , we have $|V(T)| \leq 3^{i-d} \cdot \lambda_k$ and $|N_G^\dagger[T]| \leq 3^{i-d} \cdot \lambda_k \cdot d$.*

► **Proposition 5.5** (N^\dagger Is a Destroy Neighborhood). *For all integers i, d, k , every incidence graph G in \mathcal{C}_d , every (i, d, k) -obstruction-tree T of G , and every variable x of G , if $x \notin N_G^\dagger[T]$, then T is also an (i, d, k) -obstruction-tree in at least one of $G[x_+]$ and $G[x_-]$.*

Due to space constraints, both proofs were moved to Appendix A.1 and Appendix A.2. We now turn to the main property of obstruction-trees, which explains why this notion is relevant in this context.

► **Proposition 5.6** (Obstruction-Trees Obstruct). *For all integers i, d, k and every incidence graph G in \mathcal{C}_d , if there is an (i, d, k) -obstruction-tree T of G , then $\text{srbd}_{\mathcal{C}_0}(G) \geq i$.*

Proof. Proof by induction on i .

Base Case: $i = d$. Follows immediately from Lemma 5.1.

Induction Step: Let T be an $(i + 1, d, k)$ -obstruction-tree of G , and assume towards a contradiction that $\text{srbd}_{\mathcal{C}_0}(G) < i + 1$. By Definition 5.3 we get T_1 and T_2 , two (i, d, k) -obstruction-trees connected by a path P . Additionally in every connected component H of G , $\text{srbd}_{\mathcal{C}_0}(H) < i + 1$ holds as well. Since $V(T)$ is connected, there exists one component H containing T . Then, there must exist a variable x in H such that $\text{srbd}_{\mathcal{C}_0}(H[x_+]) < i$ and $\text{srbd}_{\mathcal{C}_0}(H[x_-]) < i$. Assume $x \notin N_H^\dagger[T_1]$. Then by Proposition 5.5 we get that T_1 remains an (i, d, k) -obstruction-tree in either $H[x_+]$ or $H[x_-]$. We use the induction hypothesis to conclude that one of the graphs has strong recursive backdoor depth at least i and we get a contradiction. Assume $x \in N_H^\dagger[T_1]$. Then $x \notin N_H^\dagger[T_2]$ by Definition 5.3 and we can make the same argument. ◀

Finally, and for technical reasons, we need to show that if we assign a variable, we do not increase the strong recursive backdoor depth. Also if we find an obstruction-tree after assigning some variables, then it is also an obstruction-tree in the original graph with the same destroy neighborhood.

► **Lemma 5.7** ($\text{srbd}_{\mathcal{C}_0}$ Is Closed Under Assignments). *For every integer k , every incidence graph G , and every literal x_\star in G , if $\text{srbd}_{\mathcal{C}_0}(G) \leq k$, then $\text{srbd}_{\mathcal{C}_0}(G[x_\star]) \leq k$.*

► **Proposition 5.8** (Obstruction-Trees Can Be Lifted). *For all integers i, d, k with $d \leq i$ and $d \leq k$, every incidence graph G in \mathcal{C}_d , every obstruction-tree T and every literal x_\star of G , if T is an (i, d, k) -obstruction-tree of $H := G[x_\star]$, then it is also an (i, d, k) -obstruction-tree of G and we have $N_G^\dagger[T] = N_H^\dagger[T]$.*

The proofs of these statements can be found in Appendix A.3 and Appendix A.4.

5.2 Algorithms

We finally turn to the algorithm. We first show that we can efficiently compute an obstruction-tree, or make progress towards computing a strong recursive backdoor to \mathcal{C}_0 . Making progress here means decreasing the clause degree of the graph. At every step, the algorithm may stop if it concludes that $\text{srbd}_{\mathcal{C}_0}(G) > k$.

► **Proposition 5.9** (Obstruction-Trees Are Easy to Compute). *There is an algorithm that, given three integers i, d, k , with $d \leq i \leq k + 1$, and an incidence graph G in \mathcal{C}_d , in time $2^{2^{\mathcal{O}(k)}} \cdot |G|$ either*

1. returns an (i, d, k) -obstruction-tree T , or
2. returns a strong recursive backdoor B to \mathcal{C}_{d-1} of depth at most $g(i, d, k) := 3^{i-d} \cdot \lambda_k \cdot d$, or
3. concludes that $\text{srbd}_{\mathcal{C}_0}(G) > k$.

Proof. We fix d, k and prove the claims by induction on i and $|G|$.

Base Case: When $i = d$, we search for a clause c connected to d variables. If we find c , then c is a (d, d, k) -obstruction-tree and we return it. If there is no such clause, then G is also in \mathcal{C}_{d-1} , and the leaf node labeled G is a strong recursive backdoor to \mathcal{C}_{d-1} .

Induction Step on i : We now fix i and assume that we have a working algorithm with parameters (i, d, k) for any incidence graph $G \in \mathcal{C}_d$. We now explain by induction on $|G|$ how to build an algorithm with parameters $(i + 1, d, k)$.

Base Case: In the base case $|G| = 1$, then $G \in \mathcal{C}_0$, and there is nothing to do.

Induction Step on $|G|$ when G is not connected: All connected components of G have size strictly smaller than G , and we can run the algorithm with parameters $(i + 1, d, k)$ on each. If in one connected component H , we find an $(i + 1, d, k)$ -obstruction-tree T , then T is also an $(i + 1, d, k)$ -obstruction-tree of G and we are done. If for one connected component H we have $\text{srbd}_{\mathcal{C}_0}(H) > k$, then it also holds that $\text{srbd}_{\mathcal{C}_0}(G) > k$. Finally, if for every connected component H we find a strong recursive backdoor B_H to \mathcal{C}_{d-1} of depth at most $g(i + 1, d, k)$, we can merge them to build a recursive backdoor B to \mathcal{C}_{d-1} for G . In order to do this, we insert a root node labeled G , whose children are all the B_H . Since we do not insert a variable node, B has still depth at most $g(i + 1, d, k)$.

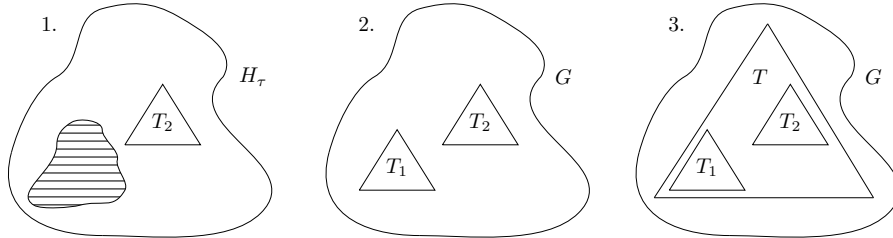
Induction Step on $|G|$ when G is connected: In this case, we use the induction hypothesis on G with parameters (i, d, k) . If the algorithm provides a strong recursive backdoor or concludes that $\text{srbd}_{\mathcal{C}_0}(G) > k$, we are done. We focus on the case where the algorithm returns an (i, d, k) -obstruction-tree T_1 for G and $N_G^\dagger[T_1]$. We define \mathcal{T} as the set of all possible truth assignments to the variables of $N_G^\dagger[T_1]$. For every τ in \mathcal{T} , we define $H_\tau := G[\tau]$. On every H_τ we run the algorithm given by induction with parameters (i, d, k) .

Case 1: For at least one H_τ we have that $\text{srbd}_{\mathcal{C}_0}(H_\tau) > k$. By Lemma 5.7, $\text{srbd}_{\mathcal{C}_0}(G) > k$ follows.

Case 2: For at least one H_τ we find an (i, d, k) -obstruction-tree T_2 . By Proposition 5.8, T_2 is also an (i, d, k) -obstruction-tree in G and $N_{H_\tau}^\dagger[T_2] = N_G^\dagger[T_2]$. Since none of the variables in $N_G^\dagger[T_1]$ appear in H_τ , we have that $N_G^\dagger[T_1] \cap N_G^\dagger[T_2] = \emptyset$. We run a BFS algorithm to find a shortest path P between a vertex of T_1 and T_2 in G . If P has length greater than λ_k , we can conclude that $\text{srbd}_{\mathcal{C}_0}(G) > k$ by Lemma 5.2. If P has length at most λ_k , we have that $T = (T_1, P, T_2)$ is an $(i + 1, d, k)$ -obstruction-tree in G .

73:12 Recursive Backdoors for SAT

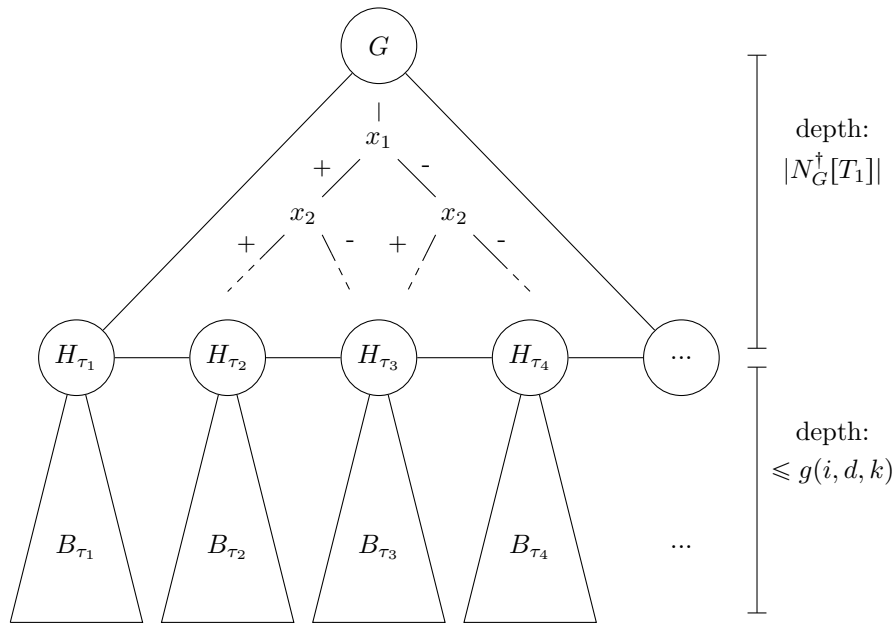
The process of constructing T is depicted in Figure 3. In the first panel we see that we have found T_2 in the graph H_τ . H_τ is an induced subgraph of G in which the variables of τ have been assigned and which therefore does not contain T_1 (see hatched area). In the second panel we lift T_2 into G , which also contains T_1 . In the third panel, all that's left to do is to combine T_1 and T_2 into T , which then is an obstruction-tree for G .



■ Figure 3 Searching an obstruction-tree.

Case 3: For every H_τ we find a strong recursive backdoor B_τ to \mathcal{C}_0 of depth at most $g(i, d, k)$. In this case, we can combine them and build a strong recursive backdoor B of G to \mathcal{C}_{d-1} . To do so, we start with the complete binary tree, where at each step we branch over one variable in $N_G^\dagger[T_1]$. At depth $|N_G^\dagger[T_1]|$, each node corresponds to an assignment τ of \mathcal{T} . We then finish the tree by plugging in B_τ in the branch corresponding to τ . B is a strong recursive backdoor of G to \mathcal{C}_{d-1} , and its depth is bounded by: $|N_G^\dagger[T_1]| + g(i, d, k) \leq g(i + 1, d, k)$.

The process of merging backdoors is depicted in Figure 4. The backdoors B_τ are depicted as trees, whose root nodes are merged by and form the leaves of a full binary tree over the variables of $N_G^\dagger[T_1]$.



■ Figure 4 Merging backdoors.

Time Complexity: The proof for the time complexity can be found in Appendix A.5. ◀

We now use the result of Proposition 5.9 sufficiently many times so that the degree of the input graph reaches 0. Again, at any point, the algorithm may stop and conclude that $\text{srbd}_{\mathcal{C}_0}(G) > k$. Remember that $\lambda_k = 4 \cdot 2^k$.

► **Theorem 5.10.** *There is an algorithm that, given as input an integer k and an incidence graph G , in time $2^{2^{\mathcal{O}(k)}} \cdot |G|$ either:*

1. *returns a strong recursive backdoor of G to \mathcal{C}_0 of depth at most $3^k \cdot \lambda_k \cdot k^2$, or*
2. *concludes that $\text{srbd}_{\mathcal{C}_0}(G) > k$.*

Proof. Let d be the maximal degree of a clause in G . If $d > k$ conclude that $\text{srbd}_{\mathcal{C}_0}(G) > k$ by Lemma 5.1. Otherwise handle G using induction on d to search for a SRB to \mathcal{C}_0 of depth at most $3^k \cdot \lambda_k \cdot d^2$:

Base Case: $G \in \mathcal{C}_0$ and the node labeled G is a SRB to \mathcal{C}_0 of depth 0.

Induction Step: $G \in \mathcal{C}_{d+1}$. Run the algorithm presented in Proposition 5.9 with parameters $(k+1, d+1, k)$ on G . If it concludes that $\text{srbd}_{\mathcal{C}_0}(G) > k$, or returns a $(k+1, d+1, k)$ -obstruction-tree, conclude that $\text{srbd}_{\mathcal{C}_0}(G) > k$ by Proposition 5.6. If a SRB B is returned, then B will have depth at most $3^{k+1-(d+1)} \cdot \lambda_k \cdot (d+1) \leq 3^k \cdot \lambda_k \cdot (d+1)$ and every leaf of B will be labeled with a graph H in \mathcal{C}_d .

We then apply the algorithm given by the induction hypothesis to every H . If for one H we get that $\text{srbd}_{\mathcal{C}_0}(H) > k$, conclude that $\text{srbd}_{\mathcal{C}_0}(G) > k$ by Lemma 5.7. If for every H we get a SRB B_H to \mathcal{C}_0 of depth at most $3^k \cdot \lambda_k \cdot d^2$, we use the results to build a SRB to \mathcal{C}_0 for G . To do so, we replace the leaf labeled H in B with B_H for every H . As a result, B will be extended to be a SRB for G to \mathcal{C}_0 with depth at most $3^k \cdot \lambda_k \cdot (d+1)^2$.

Time Complexity: The proof for the time complexity can be found in Appendix A.6. ◀

► **Corollary 5.11.** *Given a formula φ and a parameter k there is an algorithm that solves $\text{SAT}(\text{srbd}_{\mathcal{C}_0})$ in time $2^{2^{\mathcal{O}(k)}} \cdot |\varphi|$.*

Proof. We compute the satisfiability of φ in two steps. First run the algorithm given in Theorem 5.10 with parameters φ and k in time $2^{2^{\mathcal{O}(k)}} \cdot |\varphi|$. If the algorithm concludes that $\text{srbd}_{\mathcal{C}_0}(\varphi) > k$ we are finished. Otherwise a SRB with depth at most $3^k \cdot \lambda_k \cdot k^2$ of φ to \mathcal{C}_0 is returned.

Second, we make use of the calculated SRB by running the algorithm described in Proposition 3.3, to determine the satisfiability of φ . The satisfiability of a formula in \mathcal{C}_0 can be checked in constant time: If it contains no clause, then all clauses are trivially satisfied. If it contains at least one clause, then that clause is empty and unsatisfiable. Therefore the second step runs in time $\mathcal{O}(2^{3^k \cdot \lambda_k \cdot k^2} \cdot |\varphi|)$. Adding up the running times, we get a total time complexity of $2^{2^{\mathcal{O}(k)}} \cdot |\varphi|$. ◀

6 Weak Recursive Backdoor Detection to \mathcal{C}_0 Is $W[2]$ -Hard

In this section, we show that the parametrized problem of detecting a weak recursive backdoor of depth k to the class of edgeless graphs is $W[2]$ -hard when parametrized by k .

► **Theorem 6.1.** *WEAK-RECURSIVE- \mathcal{C}_0 -BACKDOOR-DETECTION is $W[2]$ -hard.*

Due to space constraints, the proof was moved to Appendix A.7.

7 Conclusion

We have proposed the new notions of *strong* and *weak recursive backdoors*, which exploit the structure of formulas that can be recursively split into independent parts by partial assignments. Recursive backdoors are measured by their depth and can contain, even at bounded depth, an unbounded number of variables. In our work we have focused on the tractable base class of empty formulas \mathcal{C}_0 . We have shown, that detecting weak recursive backdoors to \mathcal{C}_0 is $W[2]$ -hard. Our main technical contribution is an fpt algorithm that detects strong recursive backdoors of bounded depth to \mathcal{C}_0 . Even for \mathcal{C}_0 this extends tractable SAT Solving to a new class of formulas.

Our result raises the question of whether the detection of strong recursive backdoors can be expanded to larger base classes such as 2CNF or Horn. Especially 2CNF seems to be in reach, as similar to \mathcal{C}_0 , incidence graphs of formulas with bounded recursive backdoor depth to 2CNF have a bounded clause degree. This is the first ingredient for our algorithm to \mathcal{C}_0 . However our algorithm is limited to finding backdoors to \mathcal{C}_0 , as the second ingredient, which is bounded incidence graph diameter, is not given when searching for backdoors to 2CNF.

References

- 1 Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.
- 2 Marek Cygan, Fedor Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, January 2015.
- 3 Bistra Dilkina, Carla P. Gomes, and Ashish Sabharwal. Backdoors in the context of learning. In *SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 73–79. Springer, 2009.
- 4 Vijay Ganesh and Moshe Y. Vardi. On the unreasonable effectiveness of SAT solvers. In *Beyond the Worst-Case Analysis of Algorithms*, pages 547–566. Cambridge University Press, 2020.
- 5 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Backdoor treewidth for sat. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 20–37. Springer, 2017.
- 6 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. *ACM Transactions on Algorithms*, 13(2):1–32, 2017.
- 7 Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Živný. Backdoors into heterogeneous classes of SAT and CSP. *Journal of Computer and System Sciences*, 85:38–56, 2017.
- 8 Serge Gaspers and Stefan Szeider. Backdoors to satisfaction. In *The Multivariate Algorithmic Revolution and Beyond*, pages 287–317. Springer, 2012.
- 9 Serge Gaspers and Stefan Szeider. Strong backdoors to bounded treewidth SAT. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 2013.
- 10 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
- 11 N. Nishimura, P. Ragde, and Stefan Szeider. Detecting backdoor sets with respect to horn and binary clauses. In *SAT*, 2004.
- 12 Sebastian Ordyniak, André Schidler, and Stefan Szeider. Backdoor DNFs. Technical Report AC-TR-21-001, Algorithms and Complexity Group, TU Wien, 2021.
- 13 Neil Robertson and Paul D Seymour. Graph minors V. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986.
- 14 Marko Samer and Stefan Szeider. Backdoor trees. In *Proceedings of the 23rd National Conference on Artificial Intelligence – Volume 1, AAAI’08*, page 363–368. AAAI Press, 2008.

- 15 Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *Journal of Discrete Algorithms*, 8(1):50–64, 2010.
- 16 Marko Samer and Stefan Szeider. Fixed-parameter tractability. Technical Report AC-TR-21-004, Algorithms and Complexity Group, TU Wien, 2021. Chapter 17, Handbook of Satisfiability, 2nd Edition, 2021.
- 17 Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03*, page 1173–1178. Morgan Kaufmann Publishers Inc., 2003.
- 18 Edward Zulkoski, Ruben Martins, Christoph M. Wintersteiger, Robert Robere, Jia Hui Liang, Krzysztof Czarnecki, and Vijay Ganesh. Learning-sensitive backdoors with restarts. In *CP*, volume 11008 of *Lecture Notes in Computer Science*, pages 453–469. Springer, 2018.

A Omitted Proofs

A.1 Proof of Proposition 5.4

Proof. The second claim easily follows the first one. The set $N_T^\dagger[G]$ contains $\text{var}(T)$ and at most the variables connected to a clause from $\text{cla}(T)$. Since $G \in \mathcal{C}_d$, we get that $|N_G^\dagger[T]| \leq |V(T)| \cdot d$. Now we prove that $|V(T)| \leq 3^{i-d} \cdot \lambda_k$ by induction on i .

Base Case: T is an (d, d, k) -obstruction-tree. By definition, $|V(T)| = |T| = d + 1 \leq 3^{d-d} \cdot \lambda_k$.

Induction Step: T is an $(i + 1, d, k)$ -obstruction-tree. Then $T = (T_1, P, T_2)$ such that T_1 and T_2 are (i, d, k) -obstruction-trees of G and $|V(T)| \leq |V(T_1)| + |V(P)| + |V(T_2)|$. We apply our induction hypothesis to conclude that both $V(T_1)$ and $V(T_2)$ have at most $3^{i-d} \cdot \lambda_k$ elements. P has at most λ_k elements by definition. We conclude that $|V(T)| \leq 3 \cdot 3^{i-d} \cdot \lambda_k = 3^{i+1-d} \cdot \lambda_k$. ◀

A.2 Proof of Proposition 5.5

In order to prove Proposition 5.5, we first prove an intermediate result:

► **Proposition A.1** (Obstruction-Trees Are only Influenced by Adjacent Variables). *For all integers i, d, k , every incidence graph G in \mathcal{C}_d , every (i, d, k) -obstruction-tree T of G , every variable x in G , and every polarity \star , if $x \notin \text{var}(T)$ and for all $c \in \text{cla}(T)$ we have $\{x, c\} \notin E_\star$, then T is still an (i, d, k) -obstruction-tree in $G[x_\star]$.*

Proof. Proof by induction on i .

Base Case: $i = d$. Then T contains a d -clause c and its adjacent variables $\text{var}(T)$. If $x \notin \text{var}(T)$, then T remains untouched and continues to be a (d, d, k) -obstruction-tree of $G[x_\star]$.

Induction Step: $i > d$. Then $T = (T_1, P, T_2)$, where T_1 and T_2 are (i, d, k) -obstruction-trees of G . Assume $x \notin \text{var}(T)$ and for all $c \in \text{cla}(T)$ we have $\{x, c\} \notin E_\star$. Since $\text{cla}(T_1)$ and $\text{cla}(T_2)$ are both subsets of $\text{cla}(T)$ we can apply our induction hypothesis and conclude that T_1 and T_2 are still (i, d, k) -obstruction-trees of $G[x_\star]$. Since $G[x_\star]$ is a subgraph of G we know that $N_{G[x_\star]}^\dagger[T_1] \cap N_{G[x_\star]}^\dagger[T_2]$ is still empty. Since x is not contained in $\text{var}(P) \subseteq \text{var}(T)$ and is also not connected to a clause of $\text{cla}(P) \subseteq \text{cla}(T)$ by polarity \star , we conclude that P still is a path of the same length in $G[x_\star]$. It follows that T must be $(i + 1, d, k)$ -obstruction-tree in $G[x_\star]$. ◀

73:16 Recursive Backdoors for SAT

We now continue with the proof of Proposition 5.5:

Proof. Assume towards a contradiction that $x \notin N_G^\dagger[T]$ and T is no (i, d, k) -obstruction-tree of $G[x_+]$ and $G[x_-]$. If T is no (i, d, k) -obstruction-tree in $G[x_+]$, then by Proposition A.1, either $x \in \text{var}(T)$ or there exists a clause c_1 connected to x by a positive edge. Since the former contradicts with $x \notin N_G^\dagger[T]$, we have that c_1 exists. Now assume that T is also no (i, d, k) -obstruction-tree in $G[x_-]$. By the same reasoning conclude that there exists a clause c_2 connected to x by a negative edge. From the existence of both c_1 and c_2 connected with different polarities to x we conclude that $x \in N_G^\dagger[T]$ and get a contradiction. \blacktriangleleft

A.3 Proof of Lemma 5.7

Proof. Proof by induction on k .

Base Case: $k = 0$. Then G is edgeless and remains edgeless when a variable is assigned.

Induction Step: Let G be an incidence graph such that $\text{srbd}_{\emptyset_0}(G) \leq k + 1$, and let x_\star be any literal of G . If G is connected, then by Definition 3.1 there exists a variable y such that $\text{srbd}_{\emptyset_0}(G[y_\star]) \leq k$. If $x = y$, then $\text{srbd}_{\emptyset_0}(G[x_\star]) \leq k + 1$ holds trivially. If $x \neq y$ then we apply our induction hypothesis and because $\text{srbd}_{\emptyset_0}(G[y_\star]) \leq k$ get that $\text{srbd}_{\emptyset_0}(G[y_\star, x_\star]) \leq k$. This leads to $\text{srbd}_{\emptyset_0}(G[x_\star, y_\star]) \leq k$, which again implies that $\text{srbd}_{\emptyset_0}(G[x_\star]) \leq k + 1$ holds. If G contains multiple components, then the same argument applies for the component that contains x and the other components remain unchanged. \blacktriangleleft

A.4 Proof of Proposition 5.8

Proof. Proof by induction on i .

Base Case: $i = d$. Then T contains a d -clause c and its variables x_1, \dots, x_d in H and $N_H^\dagger[T]$ contains all x_i . Since G has maximal clause degree d , c must also be a d -clause in G with the same neighborhood.

Induction Step: Assume T is an $(i + 1, d, k)$ obstruction-tree of H . Then $T = (T_1, P, T_2)$ such that T_1 and T_2 are (i, d, k) -obstruction-trees of H , and P is a path of length at most λ_k . By applying the induction hypothesis, we get that T_1 and T_2 are also (i, d, k) -obstruction-trees of G and that $N_G^\dagger[T_1] = N_H^\dagger[T_1]$ and $N_G^\dagger[T_2] = N_H^\dagger[T_2]$ are disjoint. P obviously still is a path of length at most λ_k in G , so T is indeed an $(i + 1, d, k)$ -obstruction-tree of G .

We now show that $N_H^\dagger[T] = N_G^\dagger[T]$. Since H is an induced subgraph of G , we get that $N_H^\dagger[T] \subseteq N_G^\dagger[T]$. To show $N_H^\dagger[T] \supseteq N_G^\dagger[T]$, pick any variable y from $N_G^\dagger[T]$. If $y \in \text{var}(T)$ we get that $y \in N_H^\dagger[T]$ by definition. If y is positively connected to c_1 and negatively connected to c_2 for two clauses $c_1, c_2 \in \text{cla}(T)$, then $y \neq x$, since otherwise the assignment of y in H would delete a clause from T , which contradicts the fact that T is an $(i + 1, d, k)$ -obstruction-tree in H . Since y is not equal to x , its edges to c_1 and c_2 are not affected by the assignment of x in H and again $y \in N_H^\dagger[T]$ holds. It follows that $N_G^\dagger[T] = N_H^\dagger[T]$. \blacktriangleleft

A.5 Time Complexity of Proposition 5.9

Proof. Let us prove by induction that the time complexity of the algorithm presented in Proposition 5.9 is $2^{2^{O(k)}} \cdot |G|$. This clearly holds when $|G| = 1$, or when $i = d$. We now move on to the induction and analyze the run of the algorithm with parameters $(i + 1, d, k)$ on a graph G .

First, note that when we split among several connected components these components are disjoint. The sum of the sizes of these components is the size of the G . Unifying the recursive backdoor, given by the components, by adding a common root node takes at most linear time, which is consistent with our hypothesis.

Second, when the graph is connected we first run the algorithm with parameters (i, d, k) . If the run does not stop there, we have an (i, d, k) -obstruction-tree T . We then consider a number of truth assignments that is bounded by $2^{N_G^{\dagger}[T]}$. For each of these assignment, we run again our algorithm with parameters (i, d, k) , on graphs smaller than G .

If we find a second (i, d, k) -obstruction-tree we then only need to compute shortest path, which can be performed in linear time. If every truth assignment provides a backdoor-tree, plugging them together only takes time linear in the number of possible truth assignments.

All together the procedure stays linear and the constant factor gets multiplied by a factor of the form $O\left(2^{N_G^{\dagger}[T]}\right)$ each time i decreases by one. By Proposition 5.4, this is bounded by $O\left(2^{3^{i-d} \cdot \lambda_k \cdot d}\right)$. Using that both $d \leq k$ and $i \leq k + 1$, this is of the form $2^{2^{O(k)}}$. As i can decrease by one at most $i - d$ many times (and therefore at most i many times) until we get to a base case, we get that the final constant factor is of the form $\left(2^{2^{O(k)}}\right)^i = 2^{i2^{O(k)}} = 2^{2^{O(k)}}$.

We finally have that the overall complexity of a run with parameters (i, d, k) on a graph G is bounded by $2^{2^{O(k)}} \cdot |G|$. ◀

A.6 Time Complexity of Theorem 5.10

Proof. Let $f(k) \cdot |G|$ be the time complexity of Proposition 5.9 and $g(k, d)$ be $3^k \cdot \lambda_k \cdot d$. We prove the time complexity of $2^{g(k,d) \cdot d} \cdot f(k) \cdot |G|$ by induction on d . If $d = 0$ then we only have to construct a single node, which can be done in constant time. For graphs in \mathcal{C}_{d+1} , running the algorithm of Proposition 5.9 can be done in time $f(k) \cdot |G|$. If we do not find a SRB, we can abort. Otherwise we find a SRB of depth at most $g(k, d + 1)$ such that all its leaves are members of \mathcal{C}_d . We can apply our induction hypothesis and assume that for a single leaf H , we can finish in time $2^{g(k,d) \cdot d} \cdot f(k) \cdot |H|$. Since the sum of the number of vertices in all leafs of the backdoor is at most $2^{g(k,d+1)} \cdot |G|$, we get that full algorithm has a running time in

$$f(k) \cdot |G| + 2^{g(k,d) \cdot d} \cdot f(k) \cdot 2^{g(k,d+1)} \cdot |G| \leq 2^{g(k,d+1) \cdot (d+1)} \cdot f(k) \cdot |G|.$$

Since both $2^{g(k,d) \cdot d}$ and $f(k)$ are in $2^{2^{O(k)}}$, the overall time complexity of the algorithm is in $2^{2^{O(k)}} \cdot |G|$. ◀

A.7 Proof of Theorem 6.1

Proof. We are going to show the W[2]-hardness of WEAK-RECURSIVE- \mathcal{C}_0 -BACKDOOR-DETECTION (WR- \mathcal{C}_0 -BD) by reduction from the W[2]-complete SET COVER problem [2, Theorem 13.28]. An instance I of the SET COVER problem is composed of a universe U , an integer k , and a set $S \subseteq P(U)$. I is a yes-instance if there exists a subset L of S with size at most k , such that the union of all sets in L is equal to U .

We reduce $I = (S, U, k)$ to the WR- \mathcal{C}_0 -BD instance $(\varphi, k + 1)$, where φ is a CNF formula over the variables $\{b_1, \dots, b_{k+2}, s_1, \dots, s_n\}$, where $n = |S|$. This formula is constructed in the following way:

73:18 Recursive Backdoors for SAT

For each element of the universe $u \in U$ a corresponding clause σ_u is created, which we call *element-clauses*. For each set S_i a corresponding variable vertex s_i is created, which we call *set-variables*. Then s_i and σ_u are connected by a positive edge when $u \in S_i$. Therefore in the incidence graph, s_i dominates all the clauses whose corresponding elements are contained in S_i .

In addition, we create $k + 2$ fresh variables b_i . Each are individually and positively connected to a fresh clause β_i . Furthermore, all b_i are negatively connected to all σ_u , creating a $K_{k+2,|U|}$ bi-clique. More formally, we have:

$$\begin{aligned}\beta_i &:= b_i \\ \sigma_u &:= \bigvee_{i=1}^{k+2} \neg b_i \vee \bigvee_{\{i \leq n : u \in S_i\}} s_i \\ \varphi &:= \bigwedge_{i=1}^{k+2} \beta_i \wedge \bigwedge_{u \in U} \sigma_u\end{aligned}$$

We will now prove that this is in fact a valid reduction.

\Rightarrow : If $I = (\{S_1, \dots, S_n\}, U, k)$ is a yes-instance, there exists a set of indices $J \subseteq \{1, \dots, n\}$ of at most size k such that $\bigcup_{j \in J} S_j = U$. We construct the weak recursive backdoor $\{s_{j+} | j \in J\}$ of size and depth at most k that dominates all element clauses. Once every variable s_j has been assigned, every element-clause σ_u has been satisfied. Only the $k + 2$ clauses β_i remain. These clauses are disjoint. We can therefore complete the backdoor by adding at depth $k + 1$ every literal b_i simultaneously.

\Leftarrow : Let $I = (S, U, k)$ be an instance for the SET COVER, and assume that $I' = (\varphi, k)$ is a yes-instance. Then there must exist a weak recursive backdoor of depth at most k that reduces φ to an edgeless graph. In order to satisfy every β_i clause, each of the b_i literals must be contained in the backdoor. Therefore the size of the backdoor is at least $k + 2$. Since the backdoor can have depth at most $k + 1$, at least two of the β_i clauses have to be split into disconnected components and satisfied separately at some point.

Since every variable b_i is connected to every element-clause σ_u , the graph only contains one connected component, as long as a clause σ_u is not satisfied. Since the literals $\neg b_i$ cannot be part of the backdoor, there must be a set of only set-variables that when assigned satisfy every element-clause. In order to not exceed the recursion depth of $k + 1$, that set must be of size at most k . Having a set of at most k set variables satisfying every element-clause implies the existence of a set cover of U of at most k elements. So I is a yes-instance. \blacktriangleleft

Parallel Algorithms for Power Circuits and the Word Problem of the Baumslag Group

Caroline Mattes ✉

Institut für Formale Methoden der Informatik (FMI), University of Stuttgart, Germany

Armin Weiß ✉ 

Institut für Formale Methoden der Informatik (FMI), University of Stuttgart, Germany

Abstract

Power circuits have been introduced in 2012 by Myasnikov, Ushakov and Won as a data structure for non-elementarily compressed integers supporting the arithmetic operations addition and $(x, y) \mapsto x \cdot 2^y$. The same authors applied power circuits to give a polynomial-time solution to the word problem of the Baumslag group, which has a non-elementary Dehn function.

In this work, we examine power circuits and the word problem of the Baumslag group under parallel complexity aspects. In particular, we establish that the word problem of the Baumslag group can be solved in NC – even though one of the essential steps is to compare two integers given by power circuits and this, in general, is shown to be P-complete. The key observation is that the depth of the occurring power circuits is logarithmic and such power circuits can be compared in NC.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Circuit complexity

Keywords and phrases Word problem, Baumslag group, power circuit, parallel complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.74

Related Version *Full Version:* <https://arxiv.org/abs/2102.09921> [28]

Funding *Armin Weiß:* Funded by DFG project DI 435/7-1.

1 Introduction

The *word problem* of a finitely generated group G is as follows: does a given word over the generators of G represent the identity of G ? It was first studied by Dehn as one of the basic algorithmic problems in group theory [8]. Already in the 1950s, Novikov and Boone succeeded to construct finitely presented groups with an undecidable word problem [5, 33]. Nevertheless, many natural classes of groups have an (efficiently) decidable word problem – most prominently the class of linear groups (groups embeddable into a matrix group over some field): their word problem is in LOGSPACE [22, 38] – hence, in particular, in NC, i.e., decidable by Boolean circuits of polynomial size and polylogarithmic depth.

There are various other results on word problems of groups in small parallel complexity classes defined by circuits. For example the word problems of solvable linear groups are even in TC^0 (constant depth with threshold gates) [19] and the word problems of Baumslag-Solitar groups and of right-angled Artin groups are AC^0 -Turing-reducible to the word problem of a non-abelian free group [42, 18]. Moreover, Thompson’s groups are co-context-free [21] and hyperbolic groups have word problem in LOGCFL [23]. All these classes are contained within NC. On the other hand, there are also finitely presented groups with a decidable word problem but with arbitrarily high complexity [36].

A mysterious class of groups under this point of view are one-relator groups, i.e. groups that can be written as a free group modulo a normal subgroup generated by a single element (*relator*). Magnus [26] showed that one-relator groups have a decidable word problem; his algorithm is called the Magnus breakdown procedure (see also [25, 27]). Nevertheless, the



© Caroline Mattes and Armin Weiß;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 74; pp. 74:1–74:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

complexity remains an open problem – although it is not even clear whether the word problems of one-relator groups are solvable in elementary time, in [3] the question is raised whether they are actually decidable in polynomial time.

In 1969 Gilbert Baumslag defined the group $\mathbf{G}_{1,2} = \langle a, b \mid bab^{-1}a = a^2bab^{-1} \rangle$ as an example of a one-relator group which enjoys certain remarkable properties. It is infinite and non-abelian, but all its finite quotients are cyclic and, thus, it is not residually finite [4]. Moreover, Gersten showed that the Dehn function of $\mathbf{G}_{1,2}$ is non-elementary [15] and Platonov [34] made this more precise by proving that it is (roughly) $\tau(\log n)$ where $\tau(0) = 1$ and $\tau(i + 1) = 2^{\tau(i)}$ for $i \geq 0$ is the tower function (note that he calls the group Baumslag-Gersten group). Since the Dehn function gives an upper bound on the complexity of the word problem, the Baumslag group was a candidate for a group with a very difficult word problem. Indeed, when applying the Magnus breakdown procedure to an input word of length n , one obtains as intermediate results words of the form $v_1^{x_1} \cdots v_m^{x_m}$ where $v_i \in \{a, b, bab^{-1}\}$, $x_i \in \mathbb{Z}$, and $m \leq n$. The issue is that the x_i might grow up to $\tau(\log n)$; hence, this algorithm has non-elementary running time. However, as foreseen by the above-mentioned conjecture, Myasnikov, Ushakov and Won succeeded to show that the word problem of $\mathbf{G}_{1,2}$ is, indeed, decidable in polynomial time [30]. Their crucial contribution was to introduce so-called *power circuits* in [31] for compressing the x_i in the description above.

Roughly speaking, a *power circuit* is a directed acyclic graph (a dag) where the edges are labelled by ± 1 . One can define an evaluation of a vertex P as two raised to the power of the (signed) sum of the successors of P . Note that this way the value $\tau(n)$ of the tower function can be represented by an n -vertex power circuit – thus, power circuits allow for a non-elementary compression. The crucial feature for the application to the Baumslag group is that power circuits not only efficiently support the operations $+$, $-$, and $(x, y) \mapsto x \cdot 2^y$, but also the test whether $x = y$ or $x < y$ for two integers represented by power circuits can be done in polynomial time. The main technical part of the comparison algorithm is the so-called reduction process, which computes a certain normal form for power circuits.

Based on these striking results, Diekert, Laun and Ushakov [10, 9] improved the algorithm for power circuit reduction and managed to decrease the running time for the word problem of the Baumslag group from $\mathcal{O}(n^7)$ down to $\mathcal{O}(n^3)$. They also describe a polynomial-time algorithm for the word problem of the famous Higman group H_4 [16]. In [32] these algorithms have been implemented in C++. Subsequently, more applications of power circuits to these groups emerged: in [20] a polynomial time solution to the word problem in generalized Baumslag and Higman groups is given, in [12, 11] the conjugacy problem of the Baumslag group is shown to be strongly generically in P and in [2] the same is done for the conjugacy problem of the Higman group. Here “generically” roughly means that the algorithm works for most inputs (for details on the concept of generic complexity, see [17]).

Other examples where compression techniques lead to efficient algorithms in group theory can be found e.g. in [13, 14] or [24, Theorems 4.6, 4.8 and 4.9]. Finally, notice that in [29] the word search problem for the Baumslag group has been examined using parametrized complexity.

Contribution. The aim of this work is to analyze power circuits and the word problem of the Baumslag group under the view of parallel (circuit) complexity. For doing so, we first examine so-called *compact* representations of integers and show that ordinary binary representations can be converted into compact representations by constant depth circuits (i.e., in AC^0 – see Section 3). We apply this result in the power circuit reduction process, which is the main technical contribution of this paper. While [31, 10] give only polynomial

time algorithms, we present a more refined method and analyze it in terms of parametrized circuit complexity. The parameter here is the depth D of the power circuit. More precisely, we present threshold circuits of depth $\mathcal{O}(D)$ for power circuit reduction – implying our first main result:

► **Proposition A.** *The problem of comparing two integers given by power circuits of logarithmic depth is in TC^1 (decidable by logarithmic-depth, polynomial-size threshold circuits).*

We then analyze the word problem of the Baumslag group carefully. A crucial step is to show that all appearing power circuits have logarithmic depth. Using Proposition A we succeed to describe a TC^1 algorithm for computing the Britton reduction of uv if u and v are already Britton-reduced (Britton reductions are the basic step in the Magnus breakdown procedure – see Section 5 for a definition). This leads to the following result:

► **Theorem B.** *The word problem of the Baumslag group $\mathbf{G}_{1,2}$ is in TC^2 .*

In the final part of the paper we prove lower bounds on comparison in power circuits, and thus, on power circuit reduction. In particular, this emphasizes the relevance of Proposition A and shows that our parametrized analysis of power circuit reduction is essentially the best one can hope for. Moreover, Theorem C highlights the importance of the logarithmic depth bound for the power circuits appearing during the proof of Theorem B.

► **Theorem C.** *The problem of comparing two integers given by power circuits is P-complete.*

Power circuits can be seen in the broader context of arithmetic circuits and arithmetic complexity. Thus, results on power circuits also give further insight into these arithmetic circuits. Notice that the corresponding logic over natural numbers with addition and 2^x has been shown to be decidable by Semënov [37]. In the full version [28] we show that, indeed, for every power circuit with a marking M there is an arithmetic circuit of polynomial size with $+$, $-$, and 2^x -gates evaluating to the same number and vice-versa.

Due to space constraints we present only short outlines of the proofs for our main theorems; the full proofs as well as further details can be found in the full version on arXiv [28]. Details of the reduction process also can be found in the appendix.

2 Notation and preliminaries

General notions. We use standard \mathcal{O} -notation for functions from \mathbb{N} to non-negative reals $\mathbb{R}^{\geq 0}$, see e.g. [7]. Throughout, the logarithm \log is with respect to base two. The *tower function* $\tau: \mathbb{N} \rightarrow \mathbb{N}$ is defined by $\tau(0) = 1$ and $\tau(i+1) = 2^{\tau(i)}$ for $i \geq 0$. It is primitive recursive, but $\tau(6)$ written in binary cannot be stored in the memory of any conceivable real-world computer. We denote the support of a function $f: X \rightarrow \mathbb{R}$ by $\sigma(f) = \{x \in X \mid f(x) \neq 0\}$. Furthermore, the interval of integers $\{i, \dots, j\} \subseteq \mathbb{Z}$ is denoted by $[i..j]$ and we define $[n] = [0..n-1]$. We write $\mathbb{Z}[1/2] = \{p/2^q \in \mathbb{Q} \mid p, q \in \mathbb{Z}\}$ for the set of dyadic fractions.

Let Σ be a set. The set of all words over Σ is denoted by $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$. The length of a word $w \in \Sigma^*$ is denoted by $|w|$. A dag is a directed acyclic graph. For a dag Γ we write $\text{depth}(\Gamma)$ for its depth, which is the length (number of edges) of a longest path in Γ .

Complexity. We assume the reader to be familiar with the complexity classes LOGSPACE and P (polynomial time); see e.g. [1] for details. Most of the time, however, we use circuit complexity within NC.

Throughout, we assume that languages L (resp. inputs to functions f) are encoded over the binary alphabet $\{0, 1\}$. A Boolean circuit is a dag where the vertices are either input gates x_1, \dots, x_n , or NOT, AND, or OR gates. There are one or more designated output gates

and there is an order given on the output gates. All gates may have unbounded fan-in (i.e., there is no bound on the number of incoming wires). Let $k \in \mathbb{N}$. A language $L \subseteq \{0, 1\}^*$ belongs to AC^k if there exists a family $(C_n)_{n \in \mathbb{N}}$ of Boolean circuits such that $x \in L \cap \{0, 1\}^n$ if and only if the (unique) output gate of C_n evaluates to 1 when assigning $x = x_1 \cdots x_n$ to the input gates. Moreover, C_n may contain at most $n^{\mathcal{O}(1)}$ gates and have depth $\mathcal{O}(\log^k n)$. Likewise AC^k -computable functions are defined.

The class TC^k is defined analogously with the difference that also MAJORITY gates are allowed (a MAJORITY gate outputs 1 if its input contains more 1s than 0s). Moreover, $\text{NC} = \bigcup_{k \geq 0} \text{TC}^k = \bigcup_{k \geq 0} \text{AC}^k$. For more details on circuits we refer to [40]. Our algorithms (or circuits) rely on two basic building blocks which can be done in TC^0 :

► **Example 1.** Iterated addition is the following problem: on input of n binary numbers A_1, \dots, A_n each having n bits, compute $\sum_{i=1}^n A_i$. This is well-known to be in TC^0 – see e.g. [40, Theorem 1.37] for a proof.

► **Example 2.** Let $(k_1, v_1), \dots, (k_n, v_n)$ be a list of n key-value pairs (k_i, v_i) equipped with a total order on the keys k_i such that it can be decided in TC^0 whether $k_i < k_j$. Then the problem of sorting the list according to the keys is in TC^0 : the desired output is a list $(k_{\pi(1)}, v_{\pi(1)}), \dots, (k_{\pi(n)}, v_{\pi(n)})$ for some permutation π such that $k_{\pi(i)} \leq k_{\pi(j)}$ for all $i < j$.

We briefly describe a circuit family to do so: The first layer compares all pairs of keys k_i, k_j in parallel. For all i and j the next layer computes a Boolean value $P(i, j)$ which is true if and only if $|\{\ell \mid k_\ell < k_i\}| = j$. The latter is computed by iterated addition. As a final step the j -th output pair is set to (k_i, v_i) if and only if $P(i, j)$ is true.

► **Remark 3.** The class NC is contained in P if we consider uniform circuits. Roughly speaking, a circuit family is called *uniform* if the n -input circuit can be computed efficiently from the string 1^n . In order not to overload the presentation, throughout, we state all our results in the non-uniform case – all uniformity considerations are left to the reader.

Parametrized circuit complexity. In our work we also need some parametrized version of the classes TC^k , which we call *depth-parametrized* TC^k . Let $\text{par}: \{0, 1\}^* \rightarrow \mathbb{N}$ (called the *parameter*). Consider a family of circuits $(C_{n,D})_{n,D \in \mathbb{N}}$ such that $C_{n,D}$ contains at most $n^{\mathcal{O}(1)}$ gates (independently of D)¹ and has depth $\mathcal{O}(D \cdot \log^k n)$. A language L is said to be accepted by this circuit family if for all n and D and all $x \in \{0, 1\}^n$ with $\text{par}(x) \leq D$ we have $x \in L$ if and only if $C_{n,D}$ evaluates to 1 on input x . Similarly, $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ is computed by $(C_{n,D})_{n,D \in \mathbb{N}}$ if for all n and D and all $x \in \{0, 1\}^n$ with $\text{par}(x) \leq D$ the circuit $C_{n,D}$ evaluates to $f(x)$ on input x . We define DepParaTC^k as the class of languages (resp. functions) for which there are such parametrizations $\text{par}: \{0, 1\}^* \rightarrow \mathbb{N}$ and families of circuits $(C_{n,D})_{n,D \geq 0}$. Note that this is not a standard definition – but it perfectly fits our purposes.

► **Lemma 4.** Let $C > 0, k, \ell \in \mathbb{N}$ and $\text{par}: \{0, 1\}^* \rightarrow \mathbb{N}$ such that $\{w \in \{0, 1\}^* \mid \text{par}(w) \leq C \cdot \lfloor \log |w| \rfloor^\ell\} \in \text{TC}^{k+\ell}$ and $L \in \text{DepParaTC}^k$ (parametrized by par). Then $\tilde{L} = \{w \in L \mid \text{par}(w) \leq C \cdot \lfloor \log |w| \rfloor^\ell\}$ is in $\text{TC}^{k+\ell}$.

Power circuits. Consider a pair (Γ, δ) where Γ is a set of n vertices and δ is a mapping $\delta: \Gamma \times \Gamma \rightarrow \{-1, 0, +1\}$. Notice that $(\Gamma, \sigma(\delta))$ is a directed graph. Throughout we require that $(\Gamma, \sigma(\delta))$ is acyclic – i.e., it is a dag. In particular, $\delta(P, P) = 0$ for all vertices P . A

¹ Here and in most other natural applications the parameter D is bounded by the input size n . In this case, we could let the size of $C_{n,D}$ be a polynomial in both n and D – without changing the actual class.

marking is a mapping $M: \Gamma \rightarrow \{-1, 0, +1\}$. Each node $P \in \Gamma$ is associated in a natural way with a marking $\Lambda_P: \Gamma \rightarrow \{-1, 0, +1\}$, $Q \mapsto \delta(P, Q)$ called its successor marking. We define the evaluation $\varepsilon(P) \in \mathbb{R}_{>0}$ of a node ($\varepsilon(M) \in \mathbb{R}$ of a marking resp.) bottom-up in the dag by induction: leaves (nodes of out-degree 0) evaluate to 1 and, in general,

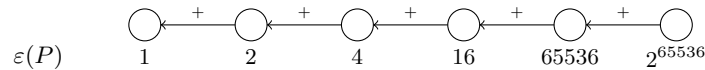
$$\varepsilon(P) = 2^{\varepsilon(\Lambda_P)} \quad \text{for a node } P, \quad \varepsilon(M) = \sum_P M(P)\varepsilon(P) \quad \text{for a marking } M.$$

► **Definition 5.** A power circuit is a pair (Γ, δ) with $\delta: \Gamma \times \Gamma \rightarrow \{-1, 0, +1\}$ such that $(\Gamma, \sigma(\delta))$ is a dag and all nodes evaluate to some positive natural number in $2^{\mathbb{N}}$.

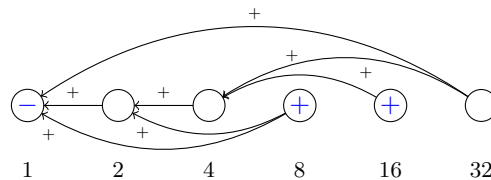
The size of a power circuit is the number of nodes $|\Gamma|$. By abuse of language, we also simply call Γ a power circuit and suppress δ whenever it is clear. If M is a marking on Γ and $S \subseteq \Gamma$, we write $M|_S$ for the restriction of M to S . Let (Γ', δ') be a power circuit, $\Gamma \subseteq \Gamma'$, $\delta = \delta'|_{\Gamma \times \Gamma}$, and $\delta'|_{\Gamma \times (\Gamma' \setminus \Gamma)} = 0$. Then (Γ, δ) itself is a power circuit. We call it a *sub-power circuit* and denote this by $(\Gamma, \delta) \leq (\Gamma', \delta')$ or, if δ is clear, by $\Gamma \leq \Gamma'$.

If M is a marking on $S \subseteq \Gamma$, we extend M to Γ by setting $M(P) = 0$ for $P \in \Gamma \setminus S$. With this convention, every marking on Γ also can be seen as a marking on Γ' if $\Gamma \leq \Gamma'$.

► **Example 6.** A power circuit of size n can realize $\tau(n)$ since a directed path of n nodes represents $\tau(n)$ as the evaluation of the last node. The following power circuit realizes $\tau(6)$ using 6 nodes:



► **Example 7.** We can represent every integer in the range $[-2^n - 1, 2^n - 1]$ as the evaluation of some marking in a power circuit with node set $\{P_0, \dots, P_{n-1}\}$ with $\varepsilon(P_i) = 2^i$ for $i \in [n]$. Thus, we can convert the binary notation of an n -bit integer into a power circuit with n vertices and $\mathcal{O}(n \log n)$ edges (each successor marking requires at most $\lfloor \log n \rfloor + 1$ edges). For an example of a marking representing the integer 23, see Figure 1.



■ **Figure 1** Each integer $z \in [-63..63]$ can be represented by a marking in the following power circuit. The marking given in blue is representing the number 23.

► **Definition 8.** We call a marking M compact if for all $P, Q \in \sigma(M)$ with $P \neq Q$ we have $|\varepsilon(\Lambda_P) - \varepsilon(\Lambda_Q)| \geq 2$. A reduced power circuit of size n is a power circuit (Γ, δ) with Γ given as a sorted list $\Gamma = (P_0, \dots, P_{n-1})$ such that all successor markings are compact and $\varepsilon(P_i) < \varepsilon(P_j)$ whenever $i < j$. In particular, all nodes have pairwise distinct evaluations.

It turns out to be crucial that the nodes in Γ are sorted by their values. Still, sometimes it is convenient to treat Γ as a set – we write $P \in \Gamma$ or $S \subseteq \Gamma$ with the obvious meaning. For more details on power circuits see [10, 31].

► **Remark 9.** If (Γ, δ) is a reduced power circuit with $\Gamma = (P_0, \dots, P_{n-1})$, we have $\delta(P_i, P_j) = 0$ for $j \geq i$. Thus, the order on Γ by evaluations is also a topological order on the dag $(\Gamma, \sigma(\delta))$.

3 Compact signed-digit representations

► **Definition 10.**

- (i) A sequence $B = (b_0, \dots, b_{m-1})$ with $b_i \in \{-1, 0, +1\}$ for $i \in [m]$ is called a signed-digit representation of $\text{val}(B) = \sum_{i=0}^{m-1} b_i \cdot 2^i \in \mathbb{Z}$.
- (ii) The digit-length of $B = (b_0, \dots, b_{m-1})$ is the maximal i with $b_{i-1} \neq 0$.
- (iii) The sequence $B = (b_0, \dots, b_{m-1})$ is called compact if $b_i b_{i-1} = 0$ for all $i \in [1..m-1]$ (i.e., no two successive digits are non-zero).

Henceforth, we abbreviate “compact signed-digit representation” with csdr. A non-negative binary number is the special case of a signed-digit representation where all b_i are 0 or 1 (note that, in general, they are not compact). In particular, every integer k can be represented as a signed-digit representation. However, in general, a signed-digit representation for an integer k is not unique. In [31, Section 2.1] a linear-time algorithm for calculating csdrs has been given; here we aim for optimizing the parallel complexity.

► **Theorem 11.** *The following is in AC^0 :*

- Input: A binary number $A = (a_0, \dots, a_{m-1})$.
Output: A compact signed-digit representation of A .

Proof sketch. Computation of the csdr is in the spirit of a carry-lookahead adder: On input of the binary number $A = (a_0, \dots, a_{m-1})$ we define

$$c_i = \bigvee_{1 \leq j \leq i} \left(a_j \wedge a_{j-1} \wedge \bigwedge_{j < k \leq i} (a_k \vee a_{k-1}) \right), \quad \text{and} \quad b_i = (a_i \oplus c_i) \cdot (-1)^{a_{i+1}}.$$

Here \oplus denotes the *exclusive or* and we treat the Boolean values 0, 1 as a subset of the integers. Then $B = (b_0, \dots, b_{m-1}, b_m)$ can be calculated in AC^0 using the above formulas. The main part of the proof consists in showing that B , indeed, is compact and that $\text{val}(B) = \text{val}(A)$. This is done by induction using the recurrence $c_0 = 0$ and $c_i = (a_i \wedge a_{i-1}) \vee (c_{i-1} \wedge (a_i \vee a_{i-1}))$ for $i \geq 1$. ◀

► **Lemma 12** ([31, Lemma 4]). *Let $A = (a_0, \dots, a_{m-1})$, $B = (b_0, \dots, b_{m-1})$ be csdrs. Then:*

- (i) $\text{val}(A) = \text{val}(B)$ if and only if $a_i = b_i$ for all $i \in [m]$.
- (ii) Assume there is some i with $a_i \neq b_i$ and let $i_0 = \max\{i \in [m] \mid a_i \neq b_i\}$. Then $\text{val}(A) < \text{val}(B)$ if and only if $a_{i_0} < b_{i_0}$.

From this lemma together with Theorem 11 it follows that each $k \in \mathbb{Z}$ can be uniquely represented by a compact signed digit representation $\text{CR}(k)$. Likewise for a signed digit representation A , we write $\text{CR}(A)$ for its compact signed digit representation.

If A and B are signed digit representations, it follows from Theorem 11 and Lemma 12 that we can calculate $\text{CR}(A)$ and $\text{CR}(A + B)$ and decide whether $\text{val}(A) < \text{val}(B)$ in AC^0 .

4 Operations on power circuits

Basic operations. Before we consider the computation of reduced power circuits, which is our main result in this section, let us introduce some more notation on power circuits and recall the basic operations from [31, 10] under circuit complexity aspects.

► **Definition 13.** *Let (Γ, δ) be a reduced power circuit with $\Gamma = (P_0, \dots, P_{n-1})$.*

- (i) A chain C of length $\ell = |C|$ in Γ starting at $P_i = \text{start}(C)$ is a sequence $(P_i, \dots, P_{i+\ell-1})$ such that $\varepsilon(P_{i+j+1}) = 2 \cdot \varepsilon(P_{i+j})$ for all $j \in [\ell - 1]$.

- (ii) We call a chain C maximal if it cannot be extended in either direction. We denote the set of all maximal chains by \mathcal{C}_Γ .
- (iii) There is a unique maximal chain C_0 containing the node P_0 of value 1. We call C_0 the initial maximal chain of Γ and denote it by $C_0 = C_0(\Gamma)$.

For an example of a power circuit with three maximal chains, see Figure 2.

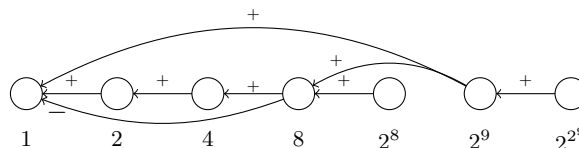


Figure 2 This power circuit is an example for a reduced power circuit with three maximal chains: The first one consists of the nodes of values 1, 2, 4, 8, the next one is formed by the nodes of values 2^8 and 2^9 and the node of value 2^{2^9} is a maximal chain of length 1.

► **Proposition 14.** Let $\Delta \in \{=, \neq, <, \leq, >, \geq\}$. The following problem is in AC^0 :

Input: A reduced power circuit (Γ, δ) with compact markings L, M and $k \in \left[0 .. \left\lfloor \frac{2^{|C_0|+1}}{3} \right\rfloor\right]$ given in binary.
Question: Is $\varepsilon(L) \Delta \varepsilon(M) + k$?

► **Lemma 15.** The following problems are all in TC^0 :

- (a) **Input:** A power circuit (Π, δ_Π) together with markings K and L .
Output: A power circuit $(\Pi', \delta_{\Pi'})$ with a marking M such that $\varepsilon(M) = \varepsilon(K) + \varepsilon(L)$ and $(\Pi, \delta_\Pi) \leq (\Pi', \delta_{\Pi'})$, $|\Pi'| \leq 2 \cdot |\Pi|$ and $\text{depth}(\Pi') = \text{depth}(\Pi)$.
- (b) **Input:** A power circuit (Π, δ_Π) together with a marking L .
Output: A marking M in the power circuit (Π, δ_Π) such that $\varepsilon(M) = -\varepsilon(L)$.
- (c) **Input:** A power circuit (Π, δ_Π) together with markings K and L such that $\varepsilon(L) \geq 0$.
Output: A power circuit $(\Pi', \delta_{\Pi'})$ with a marking M such that $\varepsilon(M) = \varepsilon(K) \cdot 2^{\varepsilon(L)}$ and $(\Pi, \delta_\Pi) \leq (\Pi', \delta_{\Pi'})$, $|\Pi'| \leq 3 \cdot |\Pi|$ and $\text{depth}(\Pi') \leq \text{depth}(\Pi) + 1$.

Lemma 15 applies the constructions from [31, Section 7] and [10, Section 2]. For (c) it can be summarized as follows: Add L to the successor marking of every node in $\sigma(K)$. To prevent other nodes from changing their value, first create disjoint copies of $\sigma(K)$ and $\sigma(L)$.

► **Remark 16.** Since membership in AC^0 often highly depends on the encoding of the input, we assume that power circuits are given in a suitable way, e.g. as an $n \times n$ matrix representing δ where each entry from $\{0, \pm 1\}$ is encoded using two bits, similarly for markings. If the power circuit is reduced, the nodes appear sorted according to their values.

We need these assumptions for proving the AC^0 -bound in Proposition 14. However, in the following, we do not consider these encoding issues because, as soon as we are dealing with TC^0 circuits, there is a lot of freedom how to encode inputs. Also note that in Lemma 15 we only state membership in TC^0 , although, with some proper work (and suitable encodings), one could also show AC^0 .

Power circuit reduction

While compact markings on a reduced power circuit yield unique representations of integers, in an arbitrary power circuit (Π, δ_Π) we can have two markings L and M such that $L \neq M$ but $\varepsilon(L) = \varepsilon(M)$. Therefore, given an arbitrary power circuit, we wish to produce a reduced power circuit for comparing markings. This is done by the following theorem, which is our main technical result on power circuits.

► **Theorem 17.** *The following is in DepParaTC^0 parametrized by $\text{depth}(\Pi)$:*

- Input:** A power circuit (Π, δ_Π) together with a marking M on Π .
Output: A reduced power circuit (Γ, δ) together with a compact marking \tilde{M} on Γ such that $\varepsilon(\tilde{M}) = \varepsilon(M)$.

For a power circuit (Π, δ_Π) with a marking M we call the power circuit (Γ, δ) together with the marking \tilde{M} obtained by Theorem 17 the *reduced form* of Π .

The proof of Theorem 17 consists of several steps, which we introduce on the next pages. The high-level idea is as follows: Like in [31, 10], we keep the invariant that there is an already reduced part and a non-reduced part (initially the non-reduced part is Π). The main difference is that in one iteration we insert *all* the nodes of the non-reduced part that have only successors in the reduced part into the reduced part. Each iteration can be done in TC^0 ; after $\text{depth}(\Pi) + 1$ iterations we obtain a reduced power circuit.

Insertion of new nodes. The following procedure is a basic tool for the reduction process. Let (Γ, δ) be a reduced power circuit and I be a set of nodes with $\Gamma \cap I = \emptyset$. Assume that for every $P \in I$ there exists a marking $\Lambda_P: \Gamma \rightarrow \{-1, 0, 1\}$ such that Λ_P is compact and $\varepsilon(\Lambda_P) \geq 0$ for all $P \in I$, and $\varepsilon(\Lambda_P) \neq \varepsilon(\Lambda_Q)$ for all $P, Q \in I \cup \Gamma$ with $P \neq Q$.

We wish to add I to the reduced power circuit (Γ, δ) . For this, we set $\Gamma' = \Gamma \cup I$ and define $\delta': \Gamma' \times \Gamma' \rightarrow \{-1, 0, 1\}$ in the obvious way: $\delta'|_{\Gamma \times \Gamma} = \delta$, $\delta'|_{\Gamma' \times I} = 0$ and $\delta'(P, Q) = \Lambda_P(Q)$ for $(P, Q) \in I \times \Gamma$. Now, (Γ', δ') is a power circuit with $(\Gamma, \delta) \leq (\Gamma', \delta')$ and for every $P \in I$ the map Λ_P is the successor marking of P . Moreover, each node of Γ' has a unique value. Since for every node $P \in \Gamma'$ the marking Λ_P is a compact marking on the reduced power circuit Γ , by Proposition 14, for $P, Q \in \Gamma'$ we are able to decide in AC^0 whether $\varepsilon(\Lambda_Q) \leq \varepsilon(\Lambda_P)$. Therefore, by Example 2 we can sort Γ' according to the values of the nodes in TC^0 and, hence, assume that $\Gamma' = (P_0, \dots, P_{|\Gamma'| - 1})$ is in increasing order. This yields the following:

► **Lemma 18 (INSERTNODES).** *The following problem is in TC^0 :*

- Input:** A power circuit (Γ, δ) and a set I with the properties described above.
Output: A reduced power circuit (Γ', δ') such that $(\Gamma, \delta) \leq (\Gamma', \delta')$ and such that for every $P \in I$ there is a node Q in Γ' with $\Lambda_Q = \Lambda_P$. In addition, $|\Gamma'| = |\Gamma| + |I|$, and $|\mathcal{C}_{\Gamma'}| \leq |\mathcal{C}_\Gamma| + |I|$.

The three steps of the reduction process. The reduction process for a power circuit (Π, δ_Π) with a marking M consists of several iterations. Each iteration starts with a power circuit $(\Gamma_i \cup \Xi_i, \delta_i)$ such that Γ_i is a reduced sub-power circuit and a marking M_i with $\varepsilon(M_i) = \varepsilon(M)$. The aim of one iteration is to integrate the vertices $\text{Min}(\Xi_i) \subseteq \Xi_i$ into Γ_i where $\text{Min}(\Xi_i)$ is defined by $\text{Min}(\Xi_i) = \{P \in \Xi_i \mid \sigma(\Lambda_P) \subseteq \Gamma_i\}$ and to update the marking M_i accordingly. Each iteration consists of the three steps **UPDATENODES**, **EXTENDCHAINS**, and **UPDATERMARKINGS**, which can be done in TC^0 . We have $\Xi_{i+1} = \Xi_i \setminus \text{Min}(\Xi_i)$. Thus, the full reduction process consists of $\text{depth}(\Pi) + 1$ many TC^0 computations. Let us now describe these three steps. The proofs of these Lemmas can be found in the appendix.

We write $(\Gamma \cup \Xi, \delta) = (\Gamma_i \cup \Xi_i, \delta_i)$ for the power circuit at the start of one iteration. Let us fix its precise properties: $\Gamma \cap \Xi = \emptyset$, $(\Gamma, \delta|_{\Gamma \times \Gamma}) \leq (\Gamma \cup \Xi, \delta)$ is a reduced power circuit and $\Lambda_P|_\Gamma$ is a compact marking for every $P \in \Xi$. Moreover, we assume that $|C_0(\Gamma)| \geq \lceil \log(|\Xi|) \rceil + 1$.

► **Lemma 19** (UPDATENODES). *The following problem is in TC^0 :*

Input: A power circuit $(\Gamma \cup \Xi, \delta)$ as above.

Output: A reduced power circuit (Γ', δ') such that $(\Gamma, \delta|_{\Gamma \times \Gamma}) \leq (\Gamma', \delta')$ and such that for every node $Q \in \text{Min}(\Xi)$ there exists a node $P \in \Gamma'$ with $\varepsilon(P) = \varepsilon(Q)$. In addition, $|\Gamma'| \leq |\Gamma| + |\text{Min}(\Xi)|$, and $|C_{\Gamma'}| \leq |C_\Gamma| + |\text{Min}(\Xi)|$.

► **Lemma 20** (EXTENDCHAINS). *The following problem is in TC^0 :*

Input: A reduced power circuit (Γ', δ') and $\mu \in \mathbb{N}$ such that $\mu \leq \lfloor \frac{2|C_0(\Gamma')|+1}{3} \rfloor$.

Output: A reduced power circuit (Γ'', δ'') such that $(\Gamma', \delta') \leq (\Gamma'', \delta'')$ and such that for each $P \in \Gamma'$ and each $i \in [0.. \mu]$ there is a node $Q \in \Gamma''$ with $\varepsilon(\Lambda_Q) = \varepsilon(\Lambda_P) + i$. In addition, $|\Gamma''| \leq |\Gamma'| + |C_{\Gamma'}| \cdot \mu$, and $|C_{\Gamma''}| \leq |C_{\Gamma'}|$.

In the following, (Γ', δ') denotes the power circuit obtained by UPDATENODES when starting with $(\Gamma \cup \Xi, \delta)$, and (Γ'', δ'') denotes the power circuit obtained by EXTENDCHAINS with $\mu = \lceil \log(|\text{Min}(\Xi)|) \rceil + 1$ on input of the power circuit (Γ', δ') (observe that, by the assumption $|C_0(\Gamma)| \geq \lceil \log(|\Xi|) \rceil + 1$, the condition on μ in Lemma 20 is satisfied). The value of μ is chosen to make sure that in the following lemma one can make the markings compact. Indeed, if $\text{Min}(\Xi) = \{P_1, \dots, P_k\}$ and all P_i have the same evaluation and are marked with 1 by M , then we might need a node of value $2^\mu \cdot \varepsilon(P_1)$ in order to make M compact.

► **Lemma 21** (UPDATERMARKINGS). *The following problem is in TC^0 :*

Input: The power circuit (Γ'', δ'') as a result of EXTENDCHAINS with $\mu = \lceil \log(|\text{Min}(\Xi)|) \rceil + 1$ and a marking M on $\Gamma \cup \Xi$.

Output: A marking \tilde{M} on $\Gamma'' \cup (\Xi \setminus \text{Min}(\Xi))$ such that $\varepsilon(\tilde{M}) = \varepsilon(M)$ and $\tilde{M}|_{\Gamma''}$ is compact.

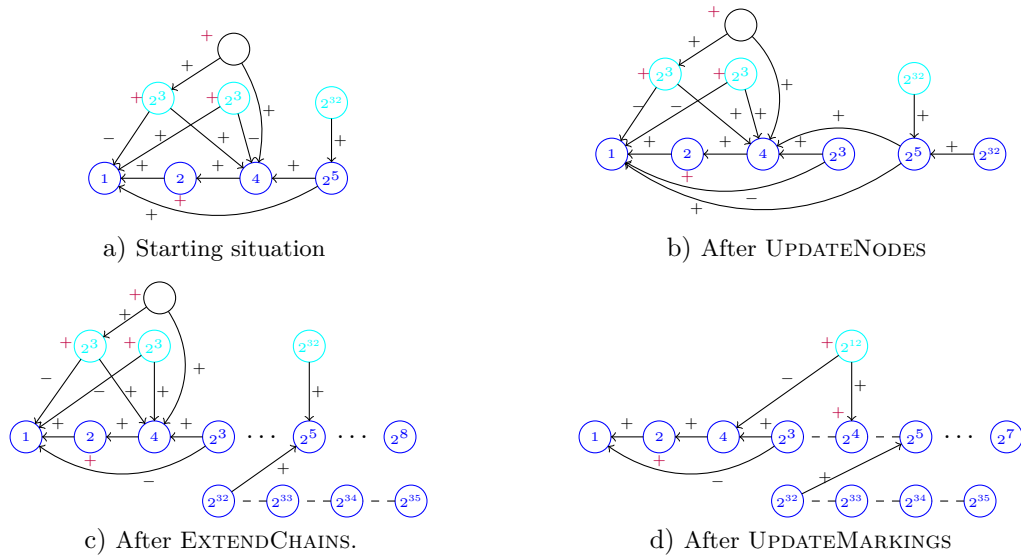
Proof sketch of Theorem 17. We start with an initial reduced power circuit (Γ_0, δ_0) (a chain of length $\lceil \log(|\Pi|) \rceil + 1$) and a non-reduced part $\Xi_0 = \Pi$ and successively apply the three steps (Lemma 19, 20, and 21) to obtain a sequence of power circuits $(\Gamma_i \cup \Xi_i, \delta_i)$ and markings M_i for $i = 0, 1 \dots$ with $\Xi_{i+1} = \Xi_i \setminus \text{Min}(\Xi_i)$ while keeping the invariants $(\Gamma_i, \delta_i|_{\Gamma_i \times \Gamma_i}) \leq (\Gamma_i \cup \Xi_i, \delta_i)$, Γ_i is reduced, $\Gamma_{i-1} \leq \Gamma_i$, $\Xi_i \subseteq \Xi_{i-1}$, and $\varepsilon(M_i) = \varepsilon(M)$. After $\text{depth}(\Pi) + 1$ iterations we reach $\Xi_{d+1} = \Xi_d \setminus \text{Min}(\Xi_d) = \emptyset$ where $d = \text{depth}(\Pi)$. Thus, $(\Gamma, \delta) = (\Gamma_{d+1}, \delta_{d+1})$ is a reduced power circuit and M_{d+1} is a compact marking on Γ_{d+1} with $\varepsilon(M_{d+1}) = \varepsilon(M)$.

▷ **Claim 22.** Let $d = \text{depth}(\Pi)$ and $\Gamma_0, \dots, \Gamma_{d+1}$ be as constructed above. Then for all i we have $|C_{\Gamma_i}| \leq |\Pi| + 1$ and $|\Gamma_i| \leq (|\Pi| + 1)^2 \cdot (\log(|\Pi|) + 2)$.

Let $D \in \mathbb{N}$ and assume that $\text{depth}(\Pi) \leq D$. By Lemma 19, 20, and 21, each iteration can be done in TC^0 . The construction of the markings \tilde{M}_i and $\tilde{\Lambda}_P$ during UPDATERMARKINGS can be done in parallel – so it is in TC^0 , although Lemma 21 is stated only for a single marking. Now, the crucial observation is that, due to Claim 22, the input size for each iteration is polynomial in the original input size of (Π, δ_Π) . Therefore, we can compose the individual iterations and obtain a circuit of polynomial size and depth bounded by $\mathcal{O}(D)$. ◀

► **Remark 23.**

(1) While Theorem 17 is only stated for one input marking, the construction works within the same complexity bounds for any number of markings on (Π, δ_Π) since during UPDATERMARKINGS these all can be updated in parallel.



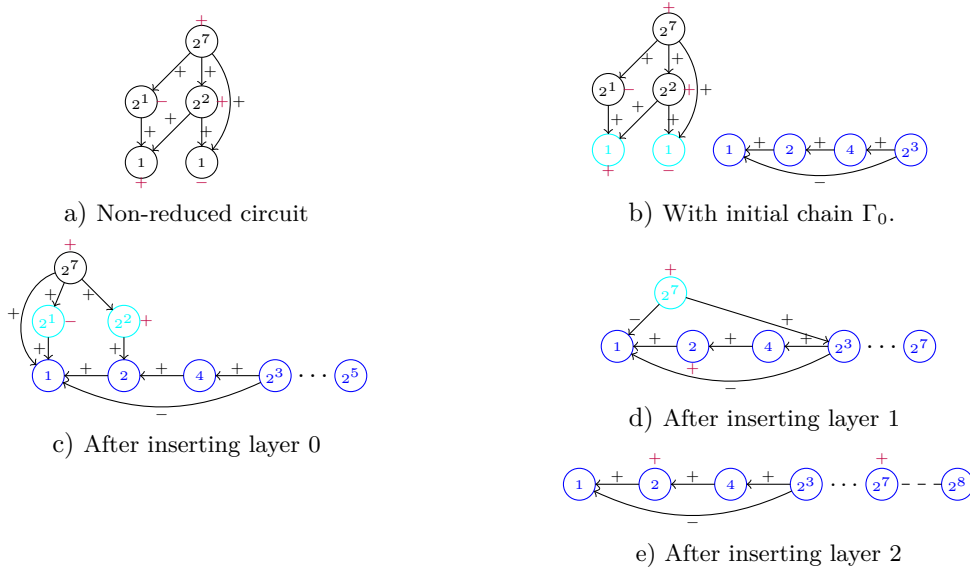
■ **Figure 3** The three steps of power circuit reduction. The already reduced part consist of **blue nodes** and $\text{Min}(\Xi_i)$ is colored in **cyan**. The **red signs** indicate a marking. Three dots \dots in between two nodes mean that we omitted some nodes. A dashed edge $--$ means that we actually omitted the outgoing edges of the right node.

- (2) Moreover, note that for every maximal chain $C \in \mathcal{C}_\Gamma$ there exists a node $Q \in \Pi$ (i.e., in the original power circuit) such that $\varepsilon(Q) = \varepsilon(\text{start}(C))$. This is because new chains are only created during UPDATENODES, the other steps only extend already existing chains.
- (3) Further observe that $|\sigma(\tilde{M})| \leq |\sigma(M)|$. Looking at the construction of \tilde{M} we see that we first make sure that M does not mark two nodes of the same value, then we make the marking compact. Both operations do not increase the number of nodes in the support of the marking.

► **Example 24.** In Figure 3 we illustrate what happens in the steps UPDATENODES, EXTENDCHAINS and UPDATEMARKINGS during the reduction process. Picture a) shows our starting situation. In b) we already inserted the nodes of value 2^3 and 2^{32} into the reduced part. Now the reduced part consists of three chains: one starting at the node of value 1 and the nodes 2^5 and 2^{32} as chains of length 1. Because $|\text{Min}(\Xi)| = 3$, we have to extend each chain by three nodes or until two chains merge. So in c) we obtain two chains, one from 1 to 2^8 and the one from 2^{32} to 2^{35} . In d) we then updated the markings and deleted the nodes from $\text{Min}(\Xi)$.

► **Example 25.** In Section 4 we give an example of the complete power circuit reduction process by showing the result after each iteration. We start with a non-reduced power circuit of depth 2 in a). This power circuit has size 5, so we first construct the starting chain of length 4 in b). Part c) and d) show the result after inserting layer 0 and layer 1, respectively. In e) we finally inserted all layers and thus have constructed the reduced power circuit.

For comparing two markings L and M on an arbitrary power circuit, we can proceed as follows: first compute the difference (Lemma 15), then reduce the power circuit (Theorem 17) and, finally, compare the resulting compact marking with zero (Proposition 14). This shows the next corollary and, together with Lemma 4, also proves Proposition A.



■ **Figure 4** The complete process of power circuit reduction – inserting layer after layer. For the meaning of the colors, see Figure 3.

► **Corollary 26.** *The following is in DepParaTC^0 parametrized by $\text{depth}(\Pi)$:*

- Input: A power circuit (Π, δ_Π) together with markings L, M on Π .
 Question: Is $\varepsilon(L) \leq \varepsilon(M)$?

Operations with floating point numbers. In the following, we want to represent a number $r \in \mathbb{Z}[1/2]$ using markings in a power circuit. For this, we use a floating point representation. Observe that for each such $r \in \mathbb{Z}[1/2] \setminus \{0\}$ there exist unique $u, e \in \mathbb{Z}$ with u odd such that $r = u \cdot 2^e$.

► **Definition 27.** *A power circuit representation of $r \in \mathbb{Z}[1/2]$ consists of a power circuit (Π, δ_Π) together with a pair of markings (U, E) on Π such that $\varepsilon(U)$ is either zero or odd and $r = \varepsilon(U) \cdot 2^{\varepsilon(E)}$.*

► **Lemma 28.** *The following problems are in DepParaTC^0 parametrized by $\text{depth}(\Pi)$:*

- Input: A power circuit representation for $r, s \in \mathbb{Z}[1/2]$ over a power circuit (Π, δ_Π) and a marking M on Π .
 Output A: A power circuit representation of $\varepsilon(M) \in \mathbb{Z}[1/2]$ over a power circuit $(\tilde{\Pi}, \delta_{\tilde{\Pi}})$.
 Output B: A power circuit representation of $r \cdot 2^{\varepsilon(M)}$ over a power circuit $(\tilde{\Pi}, \delta_{\tilde{\Pi}})$.
 Output C: A power circuit representation of $-r$ over (Π, δ_Π) .
 Output D: If $\frac{r}{s}$ is a power of two, a marking L in a power circuit $(\tilde{\Pi}, \delta_{\tilde{\Pi}})$ such that $\varepsilon(L) = \log_2(\frac{r}{s})$ (otherwise the output is undefined).
 Output E: A power circuit representation of $r + s$ over a power circuit $(\tilde{\Pi}, \delta_{\tilde{\Pi}})$.
 Output F: Is $r \in \mathbb{Z}$? If yes, a marking L in a power circuit $(\tilde{\Pi}, \delta_{\tilde{\Pi}})$ such that $\varepsilon(L) = r$.
 Question G: Is $r \triangle 0$ for $\triangle \in \{=, \neq, <, \leq, >, \geq\}$?

In all cases we have $(\Pi, \delta_\Pi) \leq (\tilde{\Pi}, \delta_{\tilde{\Pi}})$, $|\tilde{\Pi}| \in \mathcal{O}(|\Pi|)$, and $\text{depth}(\tilde{\Pi}) = \text{depth}(\Pi) + \mathcal{O}(1)$.

Proof sketch. We only outline the proof for the first point, which is the most difficult one. The other points follow rather easily using Corollary 26 and Lemma 15. Given a marking M , we wish to compute markings U, E such that $\varepsilon(M) = \varepsilon(U) \cdot 2^{\varepsilon(E)}$ and $\varepsilon(U)$ is zero or odd.

First, we construct the reduced form (Γ, δ) of Π to obtain a compact marking \tilde{M} on Γ such that $\varepsilon(M) = \varepsilon(\tilde{M}) = \sum_{i=1}^k \tilde{M}(Q_i) \cdot 2^{\varepsilon(\Lambda_{Q_i})}$ where $\sigma(\tilde{M}) = \{Q_1, \dots, Q_k\} \subseteq \Gamma$ and the Q_i are ordered according to their value. This is possible in DepParaTC^0 according to Theorem 17. It is easy to see that $|\sigma(\tilde{M})| \leq |\sigma(M)|$.

Our aim is $\varepsilon(E) = \varepsilon(\Lambda_{Q_1})$ and $\varepsilon(U) = \sum_{i=1}^k \tilde{M}(Q_i) \cdot 2^{\varepsilon(\Lambda_{Q_i}) - \varepsilon(E)}$. For this, we add nodes S_i to Π with $\varepsilon(\Lambda_{S_i}) = \varepsilon(\Lambda_{Q_i}) - \varepsilon(E)$ for $i \in [1..k]$ as follows: Looking closely at the reduction process, we can find nodes $R_i \in \Pi$ and integers $m_i \in [0..|\Gamma|]$ such that $\varepsilon(\Lambda_{Q_i}) = \varepsilon(\Lambda_{R_i}) + m_i$. To define markings M_i that evaluate to m_i , we construct nodes of depth 1 and values 2^j for $j \in [0.. \lceil \log(|\Gamma|) \rceil]$ in Π . Then $\Lambda_{S_i} = \Lambda_{R_i} + M_i - E$. So $U(S_i) = \tilde{M}(Q_i)$ for $i \in [1..k]$ and $E = \Lambda_{R_1} + M_1$ satisfies $\varepsilon(M) = \varepsilon(U) \cdot 2^{\varepsilon(E)}$. ◀

5 The word problem of the Baumslag group

Before we start solving the word problem of the Baumslag group, let us fix our notation from group theory. Let G be a group and $\eta: \Sigma^* \rightarrow G$ a surjective monoid homomorphism. We treat words over Σ both as words and as their images under η . We write $v =_G w$ with the meaning that $\eta(v) = \eta(w)$. The word problem of G is as follows: given a word $w \in \Sigma^*$, is $w =_G 1$? For further background on group theory, we refer to [25].

The Baumslag-Solitar group and the Baumslag group. The Baumslag-Solitar group is defined by $\mathbf{BS}_{1,2} = \langle a, t \mid tat^{-1} = a^2 \rangle$. We have $\mathbf{BS}_{1,2} \cong \mathbb{Z}[1/2] \rtimes \mathbb{Z}$ via the isomorphism $a \mapsto (1, 0)$ and $t \mapsto (0, 1)$. The multiplication in $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$ is defined by $(r, m) \cdot (s, n) = (r + 2^m s, m + n)$. In the following we use $\mathbf{BS}_{1,2}$ and $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$ as synonyms.

A convenient way to understand the Baumslag group $\mathbf{G}_{1,2}$ is as an HNN extension² of the Baumslag-Solitar group:

$$\mathbf{G}_{1,2} = \langle \mathbf{BS}_{1,2}, b \mid bab^{-1} = t \rangle = \langle a, t, b \mid tat^{-1} = a^2, bab^{-1} = t \rangle.$$

Note that the letter t can be seen as an abbreviation for bab^{-1} ; by removing it, we obtain exactly the presentation $\langle a, b \mid bab^{-1}a = a^2bab^{-1} \rangle$. Moreover, $\mathbf{BS}_{1,2}$ is a subgroup of $\mathbf{G}_{1,2}$ via the canonical embedding. We have $b(q, 0)b^{-1} = (0, q)$, so a conjugation by b “flips” the two components of the semi-direct product (if possible). Henceforth, we will use the alphabet $\Sigma = \{1, a, a^{-1}, t, t^{-1}, b, b^{-1}\}$ to represent elements of $\mathbf{G}_{1,2}$ (the letter 1 represents the group identity; it is there for padding reasons).

Britton reductions. Britton reductions are a standard way to solve the word problem in HNN extensions. Let $\Delta = \mathbf{BS}_{1,2} \cup \{b, b^{-1}\}$ be an infinite alphabet (note that $\Sigma \subseteq \Delta$). A word $w \in \Delta^*$ is called *Britton-reduced* if it is of the form $w = (s_0, n_0)\beta_1(s_1, n_1) \cdots \beta_\ell(s_\ell, n_\ell)$ with $\beta_i \in \{b, b^{-1}\}$ and $(s_i, n_i) \in \mathbf{BS}_{1,2}$ for all i (i.e., w does not have two successive letters from $\mathbf{BS}_{1,2}$) and there is no factor of the form $b(q, 0)b^{-1}$ or $b^{-1}(0, k)b$ with $q, k \in \mathbb{Z}$. If w is not Britton-reduced, one can apply one of the rules $(r, m)(s, n) \rightarrow (r + 2^m s, m + n)$, $b(q, 0)b^{-1} \rightarrow (0, q)$, or $b^{-1}(0, k)b \rightarrow (k, 0)$ in order to obtain a shorter word representing the same group element. The following lemma is well-known (see also [25, Section IV.2]).

► **Lemma 29** (Britton’s Lemma for $\mathbf{G}_{1,2}$ [6]). *Let $w \in \Delta^*$ be Britton-reduced. Then $w \in \mathbf{BS}_{1,2}$ as a group element if and only if w does not contain any letter b or b^{-1} . In particular, $w =_{\mathbf{G}_{1,2}} 1$ if and only if $w = (0, 0)$ or $w = 1$ as a word.*

² Named after Graham Higman, Bernhard H. Neumann and Hanna Neumann. For a precise definition, we refer to [25]. This is also the way how the Magnus breakdown procedure works.

► **Example 30.** Define words $w_0 = t$ and $w_{n+1} = b w_n a w_n^{-1} b^{-1}$ for $n \geq 0$. Then we have $|w_n| = 2^{n+2} - 3$ but $w_n =_{\mathbf{G}_{1,2}} t^{\tau(n)}$. While the length of the word w_n is only exponential in n , the length of its Britton-reduced form is $\tau(n)$.

Conditions for Britton reductions. The idea to obtain a parallel algorithm for the word problem is to compute a Britton reduction of uv given that both u and v are Britton-reduced. For this, we have to find a maximal suffix of u which cancels with a prefix of v . The following lemma is our main tool for finding the longest canceling suffix.

► **Lemma 31.** *Let $w = \beta_1(r, m)\beta_2 x \beta_2^{-1}(s, n)\beta_1^{-1} \in \Delta^*$ with $\beta_1, \beta_2 \in \{b, b^{-1}\}$ such that $\beta_1(r, m)\beta_2$ and $\beta_2^{-1}(s, n)\beta_1^{-1}$ both are Britton-reduced and $\beta_2 x \beta_2^{-1} =_{\mathbf{G}_{1,2}} (q, k) \in \mathbf{BS}_{1,2}$ (in particular, $k = 0$ and $q \in \mathbb{Z}$, or $q = 0$).*

Then $w \in \mathbf{BS}_{1,2}$ if and only if the respective condition in the following table is satisfied. Moreover, if $w \in \mathbf{BS}_{1,2}$, then $w =_{\mathbf{G}_{1,2}} \hat{w}$ according to the last column of the table.

β_1	β_2	Condition	\hat{w}
b	b	$r + 2^{m+k}s \in \mathbb{Z}, \quad m + n + k = 0$	$(0, r + 2^{-n}s)$
b	b^{-1}	$r + 2^m(q + s) \in \mathbb{Z}, \quad m + n = 0$	$(0, r + 2^m(q + s))$
b^{-1}	b	$r + 2^{m+k}s = 0$	$(n + \log(\frac{-r}{s}), 0)$
b^{-1}	b^{-1}	$r + 2^m(q + s) = 0$	$(m + n, 0)$

Notice that in the case $\beta_1 = b^{-1}$ and $\beta_2 = b$, we have $r \neq 0$ and $s \neq 0$.

► **Example 32.** Let us illustrate how to read Lemma 31 by giving an example. Let $w = \beta_1(r_1, m_1)\beta_2 x \beta_2^{-1}(s_1, n_1)\beta_1^{-1} \in \Delta^*$ with the same properties as in Lemma 31, in particular, $\beta_2 x \beta_2^{-1} =_{\mathbf{G}_{1,2}} (q, k) \in \mathbf{BS}_{1,2}$. Further assume that $\beta_1 = \beta_2 = b$. Then, according to Lemma 31, $w \in \mathbf{BS}_{1,2}$ if and only if $m_1 + n_1 = -k$ and $r_1 + 2^{m_1+k} \cdot s_1 \in \mathbb{Z}$. So we need to compute k .

Assume that $(q, k) =_{\mathbf{G}_{1,2}} \beta_2 x \beta_2^{-1} = \beta_2(r_2, m_2)\beta_3 x' \beta_3^{-1}(s_2, n_2)\beta_2^{-1}$ for some r_2, m_2, s_2, n_2 . Moreover, consider the case that $\beta_3 = b$. By applying Lemma 31 again we obtain that $(q, k) = (0, r_2 + 2^{-n_2} \cdot s_2)$. Hence, $w \in \mathbf{BS}_{1,2}$ if and only if $m_1 + n_1 + (r_2 + 2^{-n_2} \cdot s_2) = 0$ and $r_1 + 2^{m_1+r_2+2^{-n_2} \cdot s_2} \cdot s_1 \in \mathbb{Z}$. If both conditions are satisfied, then $w =_{\mathbf{G}_{1,2}} (0, r_1 + 2^{-n_1} s_1)$.

Proof sketch of Lemma 31. Consider the case that $\beta_1 = b$ and $\beta_2 = b$: Since $\beta_2 x \beta_2^{-1} \in \mathbf{BS}_{1,2}$, we have $\beta_2 x \beta_2^{-1} =_{\mathbf{G}_{1,2}} (0, k)$ for some $k \in \mathbb{Z}$. Therefore, we obtain

$$(r, m)\beta_2 x \beta_2^{-1}(s, n) =_{\mathbf{G}_{1,2}} (r, m)(0, k)(s, n) =_{\mathbf{G}_{1,2}} (r + 2^{m+k}s, m + k + n).$$

Thus, since $\beta_1 = b$, we have $w \in \mathbf{BS}_{1,2}$ if and only if $r + 2^{m+k}s \in \mathbb{Z}$ and $m + n + k = 0$. Moreover, if the latter conditions are satisfied, we have $w =_{\mathbf{G}_{1,2}} b(r + 2^{m+k}s, 0)b^{-1} = b(r + 2^{-n}s, 0)b^{-1} =_{\mathbf{G}_{1,2}} (0, r + 2^{-n}s)$. This shows the first row of the table in Lemma 31. The other rows follow with a similar calculation. ◀

Let us fix the following notation for elements $v, w \in \mathbf{G}_{1,2}$ written as words over Δ :

$$u = (r_h, m_h)\beta_h \cdots (r_1, m_1)\beta_1(r_0, m_0), \quad v = (s_0, n_0)\tilde{\beta}_1(s_1, n_1) \cdots \tilde{\beta}_\ell(s_\ell, n_\ell) \quad (1)$$

with $(r_j, m_j), (s_j, n_j) \in \mathbb{Z}[1/2] \times \mathbb{Z}$ and $\beta_j, \tilde{\beta}_j \in \{b, b^{-1}\}$. We define

$$uw[i, j] = \beta_i(r_{i-1}, m_{i-1}) \cdots \beta_1(r_0, m_0) (s_0, n_0)\tilde{\beta}_1 \cdots (s_{j-1}, n_{j-1})\tilde{\beta}_j.$$

74:14 Parallel Algorithms for the Baumslag Group

Notice that as an immediate consequence of Britton's Lemma we obtain that, if u and v as in (1) are Britton-reduced and $uv[i, i] \in \mathbf{BS}_{1,2}$ for some i , then also $uv[j, j] \in \mathbf{BS}_{1,2}$ for all $j \leq i$. Moreover, uv is Britton-reduced if and only if $\beta_1(r_0, m_0)(s_0, n_0)\beta_1 \notin \mathbf{BS}_{1,2}$.

For $\ell \in \mathbb{N}$ let \mathcal{X}_ℓ denote some set of ℓ variables. Denote by $\text{PowExp}(\mathcal{X}_\ell)$ the set of expressions which can be made up from the variables \mathcal{X}_ℓ using the operations $+$, $-$, $(r, s) \mapsto r \cdot 2^s$ if $s \in \mathbb{Z}$ (and undefined otherwise), and $(r, s) \mapsto \log(r/s)$ if $\log(r/s) \in \mathbb{Z}$ (and undefined otherwise).

► **Lemma 33.** *For every $\vec{\beta} \in \{b, b^{-1}, \perp\}^4$ there are expressions $\theta_{\vec{\beta}}, \xi_{\vec{\beta}}, \varphi_{\vec{\beta}}, \psi_{\vec{\beta}} \in \text{PowExp}(\mathcal{X}_{12})$ such that the following holds: Let $u, v \in \mathbf{G}_{1,2}$ as in (1) be Britton-reduced and assume that $uv[i-1, i-1] \in \mathbf{BS}_{1,2}$ and $\beta_i = \vec{\beta}_i^{-1}$ and let $V_i = \{r_j, s_j, m_j, n_j \mid j \in \{i-1, i-2, i-3\}\}$. If $\vec{\beta} = (\beta_i, \beta_{i-1}, \beta_{i-2}, \beta_{i-3})$ (where $\beta_j = \perp$ for $j \leq 0$), then*

1. $uv[i, i] \in \mathbf{BS}_{1,2}$ if and only if $\theta_{\vec{\beta}}(V_i) \in \mathbb{Z}$ and $\xi_{\vec{\beta}}(V_i) = 0$,
2. if $uv[i, i] \in \mathbf{BS}_{1,2}$, then $uv[i, i] =_{\mathbf{G}_{1,2}} (\varphi_{\vec{\beta}}(V_i), \psi_{\vec{\beta}}(V_i))$.

Be aware that here we have to read the set V_i of cardinality (at most) 12 as assignment to the variables \mathcal{X}_{12} . In particular, given that $uv[i-1, i-1] \in \mathbf{BS}_{1,2}$, one can decide whether $uv[i, i] \in \mathbf{BS}_{1,2}$ by looking at only constantly many letters of uv – this is the crucial observation we shall be using for describing an NC algorithm for the word problem of $\mathbf{G}_{1,2}$ (see Lemma 34 below).

Proof. W.l.o.g. $i \geq 4$. We follow the approach of Example 32. By assumption we know that there exist $q, k \in \mathbb{Z}$ such that $uv[i-1, i-1] =_{\mathbf{G}_{1,2}} (q, k) \in \mathbf{BS}_{1,2}$. According to the conditions in Lemma 31, to show Lemma 33 it suffices to find expressions $\varphi_{\vec{\beta}}(V_i), \psi_{\vec{\beta}}(V_i)$ for q and k respectively. If $(\beta_{i-1}, \beta_{i-2}) \neq (b, b^{-1})$, this follows directly from the rightmost column in Lemma 31. Otherwise, we know that $(\beta_{i-2}, \beta_{i-3}) \neq (b, b^{-1})$ and so we obtain the expressions for q and k by applying Lemma 31 to $uv[i-2, i-2]$ (note that $uv[i-2, i-2] \in \mathbf{BS}_{1,2}$ because $uv[i-1, i-1] \in \mathbf{BS}_{1,2}$). This proves the lemma. ◀

The algorithm. A power circuit representation of $u \in \mathbf{G}_{1,2}$ written as in (1) consists of the sequence $(\beta_h, \dots, \beta_1)$ and a power circuit (Π, δ_Π) with markings U_i, E_i, M_i for $i \in [0..h]$ such that (U_i, E_i) is a power circuit representation of r_i (see Definition 27) and $m_i = \varepsilon(M_i)$.

► **Lemma 34.** *The following problem is in DepParaTC^0 parametrized by $\max_i \text{depth}(\Pi_i)$:*

- Input:** Britton-reduced power circuit representations of $u, v \in \mathbf{G}_{1,2}$ over power circuits Π_1, Π_2 .
- Output:** A Britton-reduced power circuit representation of $w \in \mathbf{G}_{1,2}$ over a power circuit Π' such that $w =_{\mathbf{G}_{1,2}} uv$ and $\text{depth}(\Pi') = \max_i \text{depth}(\Pi_i) + \mathcal{O}(1)$ and $|\Pi'| \in \mathcal{O}(|\Pi_1| + |\Pi_2|)$.

Proof. Let Π be the disjoint union of Π_1 and Π_2 . We need to find the maximal i such that $uv[i, i] \in \mathbf{BS}_{1,2}$. This can be done as follows: By Lemma 28 one can evaluate the expressions $\theta_{\vec{\beta}}(V_i)$ and $\xi_{\vec{\beta}}(V_i)$ of Lemma 33 and test the conditions $\theta_{\vec{\beta}}(V_i) \in \mathbb{Z}$ and $\xi_{\vec{\beta}}(V_i) = 0$ in DepParaTC^0 . For every i this can be done independently in parallel giving us Boolean values indicating whether $uv[i-1, i-1] \in \mathbf{BS}_{1,2}$ implies $uv[i, i] \in \mathbf{BS}_{1,2}$. Now, we have to find only the maximal i_0 such that for all $j \leq i_0$ this implication is true. Since $uv[0, 0] = 1 \in \mathbf{BS}_{1,2}$, it follows inductively that $uv[i, i] \in \mathbf{BS}_{1,2}$ for all $i \leq i_0$. Moreover, as the implication $uv[i_0, i_0] \in \mathbf{BS}_{1,2} \implies uv[i_0+1, i_0+1] \in \mathbf{BS}_{1,2}$ fails, we have $uv[j, j] \notin \mathbf{BS}_{1,2}$ for $j \geq i_0+1$.

Now, using the expressions $\varphi_{\tilde{\beta}}, \psi_{\tilde{\beta}}$ from Lemma 33 one can compute again using Lemma 28 $(q, k) = (\varphi_{\tilde{\beta}}(V_{i_0}), \psi_{\tilde{\beta}}(V_{i_0})) =_{\mathbf{G}_{1,2}} uv[i_0, i_0]$ in DepParaTC^0 . Again using Lemma 28, we can compute in DepParaTC^0 $(s, m) = (r_{i_0}, m_{i_0})(q, k)(s_{i_0}, n_{i_0})$ as a power circuit representation over a power circuit $(\Pi', \delta_{\Pi'})$ with $(\Pi, \delta_{\Pi}) \leq (\Pi', \delta_{\Pi'})$, $|\Pi'| \in \mathcal{O}(\Pi)$ and $\text{depth}(\Pi') \in \text{depth}(\Pi) + \mathcal{O}(1)$. Now, the output is

$$(r_h, m_h)\beta_h \cdots (r_{i_0+1}, m_{i_0+1})\beta_{i_0+1} (s, m) \tilde{\beta}_{i_0+1}(s_{i_0+1}, n_{i_0+1}) \cdots \tilde{\beta}_\ell(s_\ell, n_\ell). \quad \blacktriangleleft$$

Instead of Theorem B, we prove the following slightly more general result. Theorem B then easily follows by Britton's Lemma. Recall that $\Sigma = \{1, a, a^{-1}, t, t^{-1}, b, b^{-1}\}$.

► **Theorem 35.** *The following problem is in TC^2 :*

Input: A word $w \in \Sigma^*$.

Output: A power circuit representation for a Britton-reduced word $w_{\text{red}} \in \Delta^*$ such that $w =_{\mathbf{G}_{1,2}} w_{\text{red}}$ and the underlying power circuit has depth $\mathcal{O}(\log |w|)$.

Proof sketch. Let $w = w_1 \cdots w_n$ with $w_j \in \Sigma$ be the input. First, we transform each letter w_j into a power circuit representation. Then, the first layer computes the Britton reduction of two-letter words using Lemma 34, the next layer takes always two of these Britton-reduced words and joins them to a new Britton-reduced word and so on. After $\log n$ layers a single Britton-reduced word remains. The crucial observation is that, due to Lemma 34, the size of the power circuits stays polynomial in n and their depth in $\mathcal{O}(\log n)$. Thus, by Lemma 4 each application of Lemma 34 is in TC^1 and the whole computation in TC^2 . \blacktriangleleft

6 P-hardness of power circuit comparison

Finally, we prove some hardness results on comparison in power circuits. In particular, they imply that Theorem 17 is optimal in a certain sense. Here, we use LOGSPACE-reductions.

► **Proposition 36.** *The following problem is NL-hard:*

Input: Given a power circuit and markings M, K .

Question: Is $\varepsilon(M) = \varepsilon(K)$?

The proof of Proposition 36 is a straightforward reduction from s - t -connectivity. For comparison with \leq , we obtain a more interesting hardness result:

► **Theorem 37.** *The following problem is P-complete:*

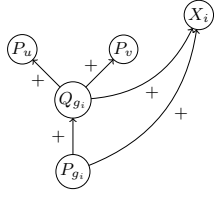
Input: A power circuit (Π, δ_{Π}) and nodes $R, S \in \Pi$ such that for all $P \in \Pi$ the marking Λ_P is compact and for all $P \neq Q$, $\varepsilon(P) \neq \varepsilon(Q)$.

Question: Is $\varepsilon(R) \leq \varepsilon(S)$?

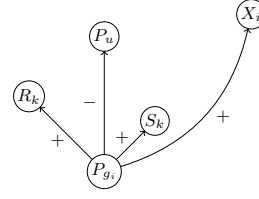
A weaker form of this result already has been stated in the second author's dissertation [41], but it never appeared in a refereed journal or conference proceedings. Notice that the only feature the power circuit in Theorem 37 lacks for being reduced is the sorting of the nodes. In particular, under the assumption $\text{NC} \neq \text{P}$, it is not possible to sort the nodes of a given power circuit in NC.

► **Remark 38.**

(a) It is an immediate consequence of Theorem 37 that the comparison problem of two markings in a power circuit is P-complete. This is because for two nodes R and S in a power circuit (Π, δ_{Π}) we have $\varepsilon(R) \leq \varepsilon(S)$ if and only if $\varepsilon(\Lambda_R) \leq \varepsilon(\Lambda_S)$.



■ **Figure 5** Power circuit for an OR gate g_i .



■ **Figure 6** Power circuit for a NOT gate g_i on level k .

- (b) If the input is given as in Theorem 37, we can check in AC^0 whether $\varepsilon(R) = \varepsilon(S)$ because this is the case if and only if $\Lambda_R(P) = \Lambda_S(P)$ for all $P \in \Gamma$ (see Lemma 12). This can be viewed as a hint that also in an arbitrary power circuit testing for equality might be easier than comparing for less than.

► **Corollary 39.** *The following problem is P-complete:*

Input: A power circuit representation of $w \in \mathbf{G}_{1,2}$.

Question: Is $w \in \mathbf{BS}_{1,2}$?

Proof sketch of Theorem 37. By [31, Proposition 49], we only need to show the hardness part. We give a reduction from the $\text{CIRCUITVALUEPROBLEM}$ which is P-complete (see [39, Thm. 10.44]). We start with a circuit \mathcal{C} of size L and depth D consisting of *input gates*, NOT gates, OR gates (of fan-in two) and one *output gate*. W.l.o.g. the circuit is layered: input gates are on level 0, and gates on level k only receive inputs from level $k - 1$. After fixing an evaluation $\text{eval}(x) \in \{0, 1\}$ for all input gates x , each gate g evaluates to a truth value $\text{eval}(g) \in \{0, 1\}$ in a natural way. The task is to compute $\text{eval}(\text{output})$. We construct a power circuit (Γ, δ) such that for every gate g on level k in \mathcal{C} there exists a node P_g in Γ satisfying

$$\begin{aligned} \tau(L - 1) < \varepsilon(\Lambda_{P_g}) &\leq \tau(2k + L) - 2 && \text{if } \text{eval}(g) = 0, \\ \tau(2k + L) &\leq \varepsilon(\Lambda_{P_g}) \leq \tau(2k + L + 1) - 2 && \text{if } \text{eval}(g) = 1. \end{aligned} \quad (2)$$

For this, we first create nodes X_k , R_k and S_k such that $\varepsilon(X_k) = 2^k$, $\varepsilon(R_k) = \tau(2k + L)$, $\varepsilon(S_k) = \tau(2k + L - 1)/2$. For an input gate g_i we set $\varepsilon(\Lambda_{P_{g_i}}) = \tau(L - 1) + i$ if $\text{eval}(g_i) = 0$ and $\varepsilon(\Lambda_{P_{g_i}}) = \tau(L) + i$ otherwise. For the *output gate* with incoming edge from gate u , we define $\varepsilon(\Lambda_{P_{\text{output}}}) = \varepsilon(P_u)$. Figure 5 and 6 illustrate the construction for OR and NOT gates. Now all nodes of Γ have pairwise different evaluations in $2^{\mathbb{N}}$ (this is essentially because we always add i to the successor marking) and compact successor markings. A rather tedious but straightforward induction shows Equation (2). Let us consider an OR gate as in Figure 5 as example: if both $\varepsilon(\Lambda_{P_u}), \varepsilon(\Lambda_{P_v}) \leq \tau(2(k - 1) + L) - 2$, then $\varepsilon(\Lambda_{P_g}) \leq 2^{2^{\varepsilon(\Lambda_{P_u})} + 2^{\varepsilon(\Lambda_{P_v})}} \leq 2^{2 \cdot 2^{\tau(2(k-1)+L)-2}} \leq \tau(2k + L) - 2$. On the other hand, if $\varepsilon(\Lambda_{P_u}) \geq \tau(2(k - 1) + L)$, then also $\varepsilon(\Lambda_{P_g}) \geq 2^{2^{\varepsilon(\Lambda_{P_u})}} \geq \tau(2k + L)$. The other cases of the induction follow similarly.

Thus, we have that $\varepsilon(P_{\text{output}}) \geq \varepsilon(R_D)$ if and only if $\text{eval}(\text{output}) = 1$. ◀

Conclusion. We showed that the word problem of the Baumslag group can be solved in TC^2 . The proof relies on the fact that all power circuits used during the execution of the algorithm have logarithmic depth. The-23 comparison problem for such power circuits is in TC^1 , although for arbitrary power circuits it is P-complete. We conclude with some open problems:

- Is it possible to reduce the complexity of the word problem of the Baumslag group any further – e.g. to find a LOGSPACE algorithm? Can we prove some non-trivial lower bounds (the word problem is NC^1 -hard as $\mathbf{G}_{1,2}$ contains a non-abelian free group [35])?
- The problem of comparing two markings on a power circuit for equality is NL-hard – is it also P-complete like comparison with less than?
- Is the word problem of the Baumslag group with power circuit representations as input P-complete? (By Corollary 39 this holds for the subgroup membership problem for $\mathbf{BS}_{1,2}$ in $\mathbf{G}_{1,2}$. Moreover, as a consequence of Proposition 36, the word problem is NL-hard.)
- By Corollary 26 for every k the comparison problem for power circuits of depth $\log^k n$ is in TC^k . Moreover, the proof of Theorem 37 can be modified to show that the same problem is hard for AC^k under AC^0 -Turing-reductions. Thus, the question remains whether, indeed, this problem is complete for TC^k under AC^0 -Turing-reductions.

References

- 1 Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009.
- 2 Owen Baker. The conjugacy problem for Higman’s group. *Internat. J. Algebra Comput.*, 30(6):1211–1235, 2020. doi:10.1142/S0218196720500393.
- 3 G. Baumslag, A. G. Myasnikov, and V. Shpilrain. Open problems in combinatorial group theory. Second Edition. In *Combinatorial and geometric group theory*, volume 296 of *Contemporary Mathematics*, pages 1–38. American Mathematical Society, 2002.
- 4 Gilbert Baumslag. A non-cyclic one-relator group all of whose finite quotients are cyclic. *Journal of the Australian Mathematical Society*, 10(3-4):497–498, 1969.
- 5 W. W. Boone. The Word Problem. *Ann. of Math.*, 70(2):207–265, 1959.
- 6 John L. Britton. The word problem. *Ann. of Math.*, 77:16–32, 1963.
- 7 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3 edition, 2009.
- 8 Max Dehn. Ueber unendliche diskontinuierliche Gruppen. *Math. Ann.*, 71:116–144, 1911.
- 9 Volker Diekert, Jörn Laun, and Alexander Ushakov. Efficient algorithms for highly compressed data: The word problem in Higman’s group is in P. In *Proc. 29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, Paris, France*, volume 14 of *LIPICs*, pages 218–229. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPICs.STACS.2012.218.
- 10 Volker Diekert, Jörn Laun, and Alexander Ushakov. Efficient algorithms for highly compressed data: The word problem in Higman’s group is in P. *International Journal of Algebra and Computation*, 22(8), 2013. doi:10.1142/S0218196712400085.
- 11 Volker Diekert, Alexei G. Myasnikov, and Armin Weiß. Conjugacy in Baumslag’s Group, Generic Case Complexity, and Division in Power Circuits. In Alberto Pardo and Alfredo Viola, editors, *Latin American Theoretical Informatics Symposium*, volume 8392 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2014. doi:10.1007/978-3-642-54423-1_1.
- 12 Volker Diekert, Alexei G. Myasnikov, and Armin Weiß. Conjugacy in Baumslag’s group, generic case complexity, and division in power circuits. *Algorithmica*, 74:961–988, 2016. doi:10.1007/s00453-016-0117-z.
- 13 Will Dison, Eduard Einstein, and Timothy R. Riley. Ackermannian integer compression and the word problem for hydra groups. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 – Kraków, Poland*, pages 30:1–30:14, 2016. doi:10.4230/LIPICs.MFCS.2016.30.
- 14 Will Dison, Eduard Einstein, and Timothy R. Riley. Taming the hydra: The word problem and extreme integer compression. *Int. J. Algebra Comput.*, 28(7):1299–1381, 2018. doi:10.1142/S0218196718500583.
- 15 S. M. Gersten. Isodiametric and isoperimetric inequalities in group extensions. Preprint, 1991.

- 16 Graham Higman. A finitely generated infinite simple group. *J. London Math. Soc.*, 26:61–64, 1951.
- 17 I. Kapovich, A. G. Miasnikov, P. Schupp, and V. Shpilrain. Generic-case complexity, decision problems in group theory and random walks. *Journal of Algebra*, 264:665–694, 2003.
- 18 Jonathan Kausch. *The parallel complexity of certain algorithmic problems in group theory*. Dissertation, Institut für Formale Methoden der Informatik, Universität Stuttgart, 2017.
- 19 Daniel König and Markus Lohrey. Evaluation of circuits over nilpotent and polycyclic groups. *Algorithmica*, 80(5):1459–1492, 2018. doi:10.1007/s00453-017-0343-z.
- 20 Jörn Laun. Efficient algorithms for highly compressed data: The word problem in generalized Higman groups is in P. *Theory Comput. Syst.*, 55(4):742–770, 2014. doi:10.1007/s00224-013-9509-5.
- 21 J. Lehnert and P. Schweitzer. The co-word problem for the Higman-Thompson group is context-free. *Bull. London Math. Soc.*, 39:235–241, 2007. doi:10.1112/blms/bd1043.
- 22 Richard J. Lipton and Yechezkel Zalcstein. Word problems solvable in logspace. *J. ACM*, 24:522–526, 1977.
- 23 Markus Lohrey. Decidability and complexity in automatic monoids. *International Journal of Foundations of Computer Science*, 16(4):707–722, 2005.
- 24 Markus Lohrey. *The Compressed Word Problem for Groups*. Springer Briefs in Mathematics. Springer, 2014. doi:10.1007/978-1-4939-0748-9.
- 25 Roger Lyndon and Paul Schupp. *Combinatorial Group Theory*. Classics in Mathematics. Springer, 2001. First edition 1977.
- 26 Wilhelm Magnus. Das Identitätsproblem für Gruppen mit einer definierenden Relation. *Mathematische Annalen*, 106:295–307, 1932.
- 27 Wilhelm Magnus, Abraham Karrass, and Donald Solitar. *Combinatorial Group Theory*. Dover, 2004.
- 28 Caroline Mattes and Armin Weiß. Parallel algorithms for power circuits and the word problem of the Baumslag group. *CoRR*, abs/2102.09921, 2021. arXiv:2102.09921.
- 29 Alexei Miasnikov and Andrey Nikolaev. On parameterized complexity of the word search problem in the Baumslag-Gersten group. In *ISSAC '20: International Symposium on Symbolic and Algebraic Computation, Kalamata, Greece, July 20-23, 2020*, pages 360–363, 2020. doi:10.1145/3373207.3404042.
- 30 Alexei G. Myasnikov, Alexander Ushakov, and Won Dong-Wook. The Word Problem in the Baumslag group with a non-elementary Dehn function is polynomial time decidable. *Journal of Algebra*, 345:324–342, 2011. URL: <http://www.sciencedirect.com/science/article/pii/S0021869311004492>.
- 31 Alexei G. Myasnikov, Alexander Ushakov, and Won Dong-Wook. Power circuits, exponential algebra, and time complexity. *International Journal of Algebra and Computation*, 22(6):3–53, 2012.
- 32 Alexei G. Myasnikov and Sasha Ushakov. Cryptography and groups (CRAG). Software Library. URL: <http://www.stevens.edu/algebraic/downloads.php>.
- 33 P. S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov*, pages 1–143, 1955. In Russian.
- 34 A. N. Platonov. Isoperimetric function of the Baumslag-Gersten group. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, 3:12–17, 2004. Russian. Engl. transl. Moscow Univ. Math. Bull. 59 (3) (2004), 12–17.
- 35 David Robinson. *Parallel Algorithms for Group Word Problems*. PhD thesis, University of California, San Diego, 1993.
- 36 Mark V. Sapir, Jean-Camille Birget, and Eliyahu Rips. Isoperimetric and Isodiametric Functions of Groups. *Ann. Math.*, 156(2):345–466, 2002.
- 37 A. L. Semenov. Logical theories of one-place functions on the natural number series. *Izv. Akad. Nauk SSSR Ser. Mat.*, 47(3):623–658, 1983.

- 38 Hans-Ulrich Simon. Word problems for groups and contextfree recognition. In *Proceedings of Fundamentals of Computation Theory (FCT'79)*, Berlin/Wendisch-Rietz (GDR), pages 417–422. Akademie-Verlag, 1979.
- 39 Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.
- 40 Heribert Vollmer. *Introduction to Circuit Complexity*. Springer, Berlin, 1999.
- 41 Armin Weiß. *On the Complexity of Conjugacy in Amalgamated Products and HNN Extensions*. Dissertation, Institut für Formale Methoden der Informatik, Universität Stuttgart, 2015.
- 42 Armin Weiß. A logspace solution to the word and conjugacy problem of generalized Baumslag-Solitar groups. In *Algebra and Computer Science*, volume 677 of *Contemporary Mathematics*, pages 185–212. American Mathematical Society, 2016.

A Details on power circuit reduction

In the following we present more details concerning the reduction process for power circuits. We give the proofs of the three steps UPDATENODES, EXTENDCHAINS, UPDATEMARKINGS and of Theorem 17. We need the following definition and lemmas. Their proofs can be found in the full version on arXiv [28].

► **Lemma 40.** *Let A be a csdr and let $B = (b_0, \dots, b_{n-1})$ be a csdr of digit-length n such that $b_i = n - i \bmod 2$ (i.e., $b_{n-1} = 1$ and then B alternates between 0 and 1). Then we have*

- (i) $\text{val}(B) = \lfloor \frac{2^{n+1}}{3} \rfloor$,
- (ii) $\text{val}(A) \leq \text{val}(B)$ if and only if the digit-length of A is at most n or $\text{val}(A) \leq 0$.

► **Definition 41.** *Let M be a marking in the reduced power circuit (Γ, δ) and let $C = (P_i, \dots, P_{i+\ell-1}) \in \mathcal{C}_\Gamma$ and define $a_j = M(P_{i+j})$ for $i \in [\ell]$. Then we write $\text{digit}_C(M) = (a_0, \dots, a_{\ell-1})$.*

► **Lemma 42.** *Let (Γ, δ) be a reduced power circuit. Let L and M be compact markings in Γ such that $\varepsilon(L) > \varepsilon(M)$ and let $0 \leq k \leq \lfloor \frac{2^{|C_0|+1}}{3} \rfloor$. Then $\varepsilon(L) \leq \varepsilon(M) + k$ if and only if $\varepsilon(M|_{\Gamma \setminus C_0}) = \varepsilon(L|_{\Gamma \setminus C_0})$ and $\varepsilon(L|_{C_0}) \leq \varepsilon(M|_{C_0}) + k$.*

For the proof of Lemma 19, we define the following equivalence relation \sim_ε on $\Gamma \cup \text{Min}(\Xi)$: $P \sim_\varepsilon Q$ if and only if $\varepsilon(P) = \varepsilon(Q)$. For $P \in \Gamma \cup \text{Min}(\Xi)$ we write $[P]_\varepsilon$ for the equivalence class containing P .

Proof of Lemma 19. Define $I \subseteq \text{Min}(\Xi)$ by taking one representative of each \sim_ε -class not containing a node of Γ . Such a set I can be computed in TC^0 : Clearly, $\text{Min}(\Xi)$ can be computed in TC^0 . The \sim_ε -classes can be computed in AC^0 by Proposition 14. Finally, for defining I one has to pick representatives, which can be done in TC^0 . Now, we can apply Lemma 18 to insert I into Γ in TC^0 . This yields our power circuit (Γ', δ') . The size bounds follow now immediately from those in Lemma 18. ◀

Proof of Lemma 20. First assume that $|C_0| = 1$. Then $|\Gamma'| = 1$ and $\mu \leq 1$. If $\mu = 1$, then just one node has to be created, namely the one of value 2 and we are done. Thus, in the following we can assume that $|C_0| \geq 2$. Now, the proof of Lemma 20 consists of two steps: first, we extend only the chain C_0 to some longer (and long enough) chain in order to make sure that the values of the (compact) successor markings of the nodes we wish to introduce can be represented within the power circuit; only afterwards we add the new nodes as described in the lemma.

Step 1: We first want to extend the chain C_0 to the chain \tilde{C}_0 of minimal length such that \tilde{C}_0 is a maximal chain, $C_0 \subseteq \tilde{C}_0$, and the last node of \tilde{C}_0 is not already present in Γ' . The resulting power circuit will be denoted by $\tilde{\Gamma}$. We define

$$i_0 = \min \{i \in [|\Gamma'|] \mid \varepsilon(\Lambda_{P_{i+1}}) - \varepsilon(\Lambda_{P_i}) > 2\}.$$

We use the convention that $P_{|\Gamma'|}$ has value infinity, so i_0 indeed exists. Furthermore, we define

$$I = \{i \in [0..i_0] \mid \varepsilon(\Lambda_{P_{i+1}}) - \varepsilon(\Lambda_{P_i}) \geq 2\}.$$

Thus, in order to obtain $\tilde{\Gamma}$, we need to insert a new node between P_i and P_{i+1} into Γ' for each $i \in I$ (resp. one node above P_{i_0}). Since the successor markings of these new nodes might point to some of the other new nodes, we cannot apply Lemma 18 as a black-box. Instead, we need to take some more care: the rough idea is that, first, we compute all positions I where new nodes need to be introduced (I is as defined above), then we compute csdrs for the respective successor markings, and, finally, we introduce these new nodes all at once knowing that all nodes where the successor markings point to are also introduced at the same time. In order to map the positions of nodes in Γ' to positions of nodes in $\tilde{\Gamma}$, we introduce a function $\lambda: [|\Gamma'|] \rightarrow \mathbb{N}$ with $\lambda(i) = i + |I \cap [0..i-1]|$.

Observe that $\lambda(i) = i$ for $i \in [C_0]$, and $\lambda(i+1) = \lambda(i) + 2$ for $i \in I$, and $\lambda(j) = j + |I|$ for $j \geq i_0 + 1$.

For each $i \in I$ we introduce a node Q_i whose successor marking we will specify later such that $\varepsilon(Q_i) = 2\varepsilon(P_i)$. We define the new power circuit $\tilde{\Gamma} = (\tilde{P}_0, \dots, \tilde{P}_{|\Gamma'|+|I|-1})$ by

$$\tilde{P}_j = \begin{cases} P_i & \text{if } j = \lambda(i) \\ Q_i & \text{if } j = \lambda(i) + 1 \text{ and } i \in I. \end{cases}$$

Notice that, if $j = \lambda(i) + 1$ for some $i \in I$, then $j \neq \lambda(i)$ for any i – hence, \tilde{P}_j is well-defined in any case.

The nodes $\tilde{P}_0, \dots, \tilde{P}_{\lambda(i_0)+1}$ will form the chain \tilde{C}_0 as claimed above. Moreover, we have $\Gamma' \subseteq \tilde{\Gamma}$ and $\tilde{\Gamma}$ is sorted increasingly. The successor markings of nodes from Γ' remain unchanged (i.e., $\Lambda_{\tilde{P}_{\lambda(i)}}(\tilde{P}_{\lambda(j)}) = \Lambda_{P_i}(P_j)$ for $i, j \in [|\Gamma'|]$ and $\Lambda_{\tilde{P}_{\lambda(i)}}(Q_j) = 0$ for $j \in I$).

For every $i \in I$ we define the successor marking of the node Q_i by

$$\text{digit}_{\tilde{C}_0}(\Lambda_{Q_i}) = \text{CR}(\varepsilon(\Lambda_{P_i}) + 1) \quad \text{and} \quad \Lambda_{Q_i}|_{\tilde{\Gamma} \setminus \tilde{C}_0} = 0.$$

Be aware that, since $Q_i \in \tilde{C}_0$, also the successor marking of Q_i (of value $\varepsilon(\Lambda_{P_i}) + 1$) can be represented using only the nodes from \tilde{C}_0 (see Remark 9), so this is, indeed, a meaningful definition (be aware that to represent $\varepsilon(\Lambda_{P_i}) + 1$, we might need some of the additional nodes Q_i , but never a node that is not part of the chain \tilde{C}_0). Clearly, this yields $\varepsilon(\Lambda_{Q_i}) = \varepsilon(\Lambda_{P_i}) + 1$ as desired.

We obtain a reduced power circuit $(\tilde{\Gamma}, \tilde{\delta})$ with $(\Gamma', \delta') \leq (\tilde{\Gamma}, \tilde{\delta})$ where the map $\tilde{\delta}: \tilde{\Gamma} \rightarrow \{-1, 0, 1\}$ is defined by the successor markings. Moreover, $\tilde{C}_0 \subseteq \tilde{\Gamma}$ has the required properties.

It remains to show that $\tilde{\Gamma}$ can be computed in TC^0 : As $|C_0| \geq 2$, according to Proposition 14, we are able to decide in AC^0 whether the markings Λ_{P_i} and $\Lambda_{P_{i+1}}$ differ by 1, 2, or more than 2 – for all $i \in [|\Gamma'|]$ in parallel. Now, i_0 can be determined in TC^0 via its definition as above. Likewise I and the function λ can be computed in TC^0 . Using Theorem 11, $\text{CR}(\varepsilon(\Lambda_{P_i}) + 1)$ for $i \in I$ can be computed in AC^0 (since $|\tilde{C}_0| \leq 2 \cdot |\Gamma'|$) showing that altogether $\tilde{\Gamma}$ can be computed in TC^0 .

Step 2: The second step is to add nodes above each chain of $\tilde{\Gamma}$ as required in the Lemma. The outcome will be denoted by (Γ'', δ'') . We start by defining

$$\begin{aligned} d_i &= \min\{\varepsilon(\Lambda_{\tilde{P}_{i+1}}) - \varepsilon(\Lambda_{\tilde{P}_i}) - 1, \mu\} && \text{for } i \in [|\tilde{\Gamma}|] \setminus \{|\tilde{C}_0| - 1\} \quad \text{and} \\ d_i &= \min\{\varepsilon(\Lambda_{\tilde{P}_{i+1}}) - \varepsilon(\Lambda_{\tilde{P}_i}) - 1, \mu - 1\} && \text{for } i = |\tilde{C}_0| - 1. \end{aligned}$$

In order to obtain (Γ'', δ'') from $(\tilde{\Gamma}, \tilde{\delta})$, for every $i \in [|\tilde{\Gamma}|]$ and every $h \in [1..d_i]$ we have to insert a node $R^{(i,h)}$ such that $\varepsilon(\Lambda_{R^{(i,h)}}) = \varepsilon(\Lambda_{\tilde{P}_i}) + h$. Observe that the numbers d_i can be computed in TC^0 : since

$$\mu + 1 \leq \left\lfloor \frac{2^{|\tilde{C}_0|+1}}{3} \right\rfloor + 1 \leq \left\lfloor \frac{2^{|\tilde{C}_0|}}{3} \right\rfloor + 1 \leq \left\lfloor \frac{2^{|\tilde{C}_0|+1}}{3} \right\rfloor,$$

by Proposition 14, we can check in AC^0 whether $\varepsilon(\Lambda_{\tilde{P}_{i+1}}) \leq \varepsilon(\Lambda_{\tilde{P}_i}) + k$ with $k \leq \mu + 1$. If $i = |\tilde{C}_0| - 1$ we choose $k = \mu$, otherwise $k = \mu + 1$. If the respective inequality holds, we obtain by Lemma 42 that $\varepsilon(\Lambda_{\tilde{P}_{i+1}}) - \varepsilon(\Lambda_{\tilde{P}_i}) - 1 = \varepsilon(\Lambda_{\tilde{P}_{i+1}}|_{\tilde{C}_0}) - \varepsilon(\Lambda_{\tilde{P}_i}|_{\tilde{C}_0}) - 1$. For the latter we have signed-digit representations of digit-length at most $|\tilde{C}_0|$. Hence, this difference can be computed in TC^0 .

Since $\tilde{P}_{|\tilde{C}_0|-1} \notin \Gamma'$ and in Step 1 we have not introduced any vertex above $\tilde{P}_{|\tilde{C}_0|-1}$, we know that $\tilde{P}_{|\tilde{C}_0|-1}$ is not marked by $\Lambda_{\tilde{P}}$ for any $\tilde{P} \in \tilde{\Gamma}$. Therefore, for all $i \in [|\tilde{\Gamma}|]$ we have $\varepsilon(\Lambda_{\tilde{P}_i}|_{\tilde{C}_0}) + \mu \leq \left\lfloor \frac{2^{|\tilde{C}_0|}}{3} \right\rfloor + \left\lfloor \frac{2^{|\tilde{C}_0|+1}}{3} \right\rfloor \leq 2 \left\lfloor \frac{2^{|\tilde{C}_0|}}{3} \right\rfloor$ and, hence, by Lemma 40, $\varepsilon(\Lambda_{\tilde{P}_i}|_{\tilde{C}_0}) + h$ can be represented as a compact marking using only nodes from \tilde{C}_0 for every $h \in [1..d_i]$. Thus, for every $d_i \neq 0$ and every $h \in [1..d_i]$ we define a successor marking of $R^{(i,h)}$ by

$$\text{digit}_{\tilde{C}_0}(\Lambda_{R^{(i,h)}}) = \text{CR}(\varepsilon(\Lambda_{\tilde{P}_i}|_{\tilde{C}_0}) + h) \quad \text{and} \quad \Lambda_{R^{(i,h)}}|_{\tilde{\Gamma} \setminus \tilde{C}_0} = \Lambda_{\tilde{P}_i}|_{\tilde{\Gamma} \setminus \tilde{C}_0}.$$

Again, we know that $|\tilde{C}_0| \leq 2|\tilde{\Gamma}'|$. With Theorem 11 we are able to calculate $\text{CR}(\varepsilon(\Lambda_{\tilde{P}_i}|_{\tilde{C}_0}) + h)$ in AC^0 .

Now we set $I = \{R^{(i,h)} \mid d_i \neq 0, h \in [1..d_i]\}$. According to Lemma 18 we are able to construct in TC^0 a reduced power circuit (Γ'', δ'') such that $(\tilde{\Gamma}, \tilde{\delta}) \leq (\Gamma'', \delta'')$ and such that for each $R \in I$ there exists a node $Q \in \Gamma''$ with $\varepsilon(Q) = \varepsilon(R)$.

Considering the size of Γ'' , observe that during the whole construction, for every node $P_i \in \Gamma'$ we create at most μ new nodes between P_i and P_{i+1} . Moreover, we only create new nodes between P_i and P_{i+1} if P_i is the last node of a maximal chain of Γ' . Furthermore, notice that the only node of Γ' above which we have introduced new nodes in both Step 1 and Step 2 is the second largest node of \tilde{C}_0 : in Step 1 we have created one new node and in Step 2 we have created at most $\mu - 1$ new nodes above it. Thus, for every chain of Γ' we have introduced at most μ new nodes. Thus, $|\Gamma''| \leq |\Gamma'| + |\mathcal{C}_{\Gamma'}| \cdot \mu$. Finally, the new nodes we create only prolongate the already existing chains, so we do not create any new chains. This finishes the proof of the lemma. \blacktriangleleft

Proof of Lemma 21. Consider again the equivalence relation \sim_ε as defined above on $\Gamma'' \cup \text{Min}(\Xi)$. For the equivalence class of a node $P \in \Gamma'' \cup \text{Min}(\Xi)$ we write $[P]_\varepsilon$. We will define the marking \tilde{M} on Γ'' by defining it on each maximal chain. Recall that we can view M as a marking on $\Gamma'' \cup \Xi$ by defining $M(P) = 0$ if $P \notin \Gamma \cup \Xi$.

Let $C = (P_i, \dots, P_{i+h-1}) \in \mathcal{C}_{\Gamma''}$ be a maximal chain of length h and let

$$S = \bigcup_{P \in C} [P]_\varepsilon = \bigcup_{P \in C} \{Q \in \Gamma'' \cup \text{Min}(\Xi) \mid \varepsilon(Q) = \varepsilon(P)\} \subseteq \Gamma'' \cup \text{Min}(\Xi).$$

74:22 Parallel Algorithms for the Baumslag Group

We wish to find a compact marking \tilde{M}_C with support contained in $C \subseteq \Gamma''$ and evaluation $\varepsilon(\tilde{M}_C) = \varepsilon(M|_S)$. First define the integer

$$Z_{M,C} = \sum_{r=0}^{h-1} \left(\sum_{Q \in [P_{i+r}]_\varepsilon} M(Q) \right) 2^r.$$

Then we have

$$\begin{aligned} Z_{M,C} \cdot \varepsilon(\text{start}(C)) &= \sum_{r=0}^{h-1} \sum_{Q \in [P_{i+r}]_\varepsilon} M(Q) 2^r \cdot \varepsilon(\text{start}(C)) \\ &= \sum_{Q \in S} M(Q) \varepsilon(Q) = \varepsilon(M|_S). \end{aligned}$$

Thus, defining \tilde{M}_C by $\text{digit}_C(\tilde{M}_C) = \text{CR}(Z_{M,C})$ gives our desired marking.

However, be aware that, for this, we have to show that the digit-length of $\text{CR}(Z_{M,C})$ is at most $|C| = h$. Let k be maximal such that $P_{i+k} \in \Gamma'$. Then, in particular, no node in S with higher evaluation than P_{i+k} is marked by M . Moreover, by the properties of $\text{EXTENDCHAINS}(\lceil \log(|\text{Min}(\Xi)|) \rceil + 1)$, we have $h - 1 - k \geq \lceil \log(|\text{Min}(\Xi)|) \rceil + 1$. Therefore,

$$\begin{aligned} Z_{M,C} &\leq \text{val}(\text{digit}_C(M)) + |\text{Min}(\Xi)| \cdot 2^k \\ &\leq \frac{1}{3} \cdot 2^{k+2} + 2^{k+\log(|\text{Min}(\Xi)|)} && \text{(by Lemma 40)} \\ &\leq \frac{4}{3} \cdot \left(2^k + 2^{k+\log(|\text{Min}(\Xi)|)} \right) \\ &\leq \frac{2}{3} \cdot 2^{k+\lceil \log(|\text{Min}(\Xi)|) \rceil + 2}. \end{aligned}$$

Thus, by Lemma 40, the digit-length of $\text{CR}(Z_{M,C})$ is at most $k + \lceil \log(|\text{Min}(\Xi)|) \rceil + 2 \leq h$. As an easy consequence of Proposition 14, the maximal chains can be determined in TC^0 . Now, for every maximal chain C the (binary) number $Z_{M,C}$ can be computed in TC^0 using iterated addition and made compact in AC^0 using Theorem 11. Thus, the marking \tilde{M}_C can be computed in TC^0 . The marking \tilde{M} as desired in the lemma is simply defined by $\tilde{M}|_{\Xi \setminus \text{Min}(\Xi)} = M|_{\Xi \setminus \text{Min}(\Xi)}$ and $\tilde{M}|_C = \tilde{M}_C|_C$ for $C \in \mathcal{C}_{\Gamma''}$ – all the markings \tilde{M}_C can be computed in parallel. \blacktriangleleft

Proof of Theorem 17. Now we are ready to describe the full reduction process based on the three steps described above. We aim for a DepParaTC^0 circuit where the input is parametrized by the depth of the power circuit. The input is some arbitrary power circuit (Π, δ_Π) together with a marking M on Π . We start with some initial reduced power circuit (Γ_0, δ_0) and some non-reduced part $\Xi_0 = \Pi$ and successively apply the three steps to obtain power circuits $(\Gamma_i \cup \Xi_i, \delta_i)$ and markings M_i for $i = 0, 1, \dots$ while keeping the following invariants:

- $(\Gamma_i, \delta_i|_{\Gamma_i \times \Gamma_i}) \leq (\Gamma_i \cup \Xi_i, \delta_i)$ (i.e., there are no edges from Γ_i to Ξ_i),
- Γ_i is reduced,
- $\Gamma_{i-1} \leq \Gamma_i$ and $\Xi_i \subseteq \Xi_{i-1}$,
- $\varepsilon(M_i) = \varepsilon(M)$.

Moreover, as long as $\Xi_{i-1} \neq \emptyset$ we assure that $\text{depth}(\Xi_i) < \text{depth}(\Xi_{i-1})$.

We first construct the initial reduced power circuit $(\Gamma_0, \tilde{\delta}_0)$ which consists exactly of a chain of length $\ell = \lceil \log(|\Pi|) \rceil + 1$. This can be done as follows: Let $\Gamma_0 = (P_0, \dots, P_{\ell-1}) = C_0$ and define successor markings by $\text{digit}_{C_0}(\Lambda_{P_i}) = \text{CR}(i)$ for $i \in [\ell]$. This defines $\tilde{\delta}_0$. Now we set $\Xi_0 = \Pi$ and we define $\delta_0: (\Gamma_0 \cup \Xi_0) \times (\Gamma_0 \cup \Xi_0) \rightarrow \{-1, 0, 1\}$ by $\delta_0|_{\Gamma_0 \times \Gamma_0} = \tilde{\delta}_0$,

$\delta_0|_{\Xi_0 \times \Xi_0} = \delta_\Pi$ and $\delta = 0$ otherwise. We extend the marking M to Γ_0 by setting $M(P) = 0$ for all $P \in \Gamma_0$. So we obtain a power circuit of the form $(\Gamma_0 \cup \Xi_0, \delta_0)$ with the properties described above.

Now let the power circuit $(\Gamma_i \cup \Xi_i, \delta_i)$ together with the marking M_i be the input for the $i + 1$ -th iteration meeting the above described invariants. We write $\tilde{\delta}_i = \delta_i|_{\Gamma_i \times \Gamma_i}$. Now we apply the three steps from above:

1. Using UPDATENODES (Lemma 19) we compute a reduced power circuit (Γ'_i, δ'_i) with $(\Gamma_i, \tilde{\delta}_i) \leq (\Gamma'_i, \delta'_i)$ such that for every $P \in \text{Min}(\Xi_i)$ there is some $Q \in \Gamma'_i$ with $\varepsilon(Q) = \varepsilon(P)$.
2. Using EXTENDCHAINS (Lemma 20) with $\mu = \lceil \log(|\text{Min}(\Xi_i)|) \rceil + 1$ we extend each maximal chain in (Γ'_i, δ'_i) by at most $\lceil \log(|\text{Min}(\Xi_i)|) \rceil + 1$ nodes. Notice that $\lceil \log(|\text{Min}(\Xi_i)|) \rceil + 1 \leq \lceil \log(|\Pi|) \rceil + 1$ and so, as $\Gamma_0 \leq \Gamma'_i$, the condition $\mu \leq \left\lfloor \frac{2^{\lceil C_0(\Gamma'_i) \rceil + 1}}{3} \right\rfloor$ in Lemma 20 is satisfied. The result of this step is denoted by (Γ''_i, δ''_i) .
3. We apply UPDATEMARKINGS (Lemma 21) to obtain markings \tilde{M}_i and $\tilde{\Lambda}_P$ for $P \in \Xi_i \setminus \text{Min}(\Xi_i)$ on $\Gamma''_i \cup (\Xi_i \setminus \text{Min}(\Xi_i))$ such that $\varepsilon(\tilde{M}_i) = \varepsilon(M_i)$ and $\varepsilon(\tilde{\Lambda}_P) = \varepsilon(\Lambda_P)$. Observe that these markings restricted to Γ''_i are compact.
4. Each iteration ends by setting $\Gamma_{i+1} = \Gamma''_i$ and $\Xi_{i+1} = \Xi_i \setminus \text{Min}(\Xi_i)$ and $M_{i+1} = \tilde{M}_i$. Finally, δ_{i+1} is defined as δ''_i on Γ_{i+1} and via the successor markings $\tilde{\Lambda}_P$ for $P \in \Xi_{i+1}$.

After exactly $\text{depth}(\Pi) + 1$ iterations we reach $\Xi_{d+1} = \Xi_d \setminus \text{Min}(\Xi_d) = \emptyset$ where $d = \text{depth}(\Pi)$. In this case we do not change the resulting power circuit any further. It is clear from Lemma 19, Lemma 20 and Lemma 21 that throughout the above-mentioned invariants are maintained. Thus, $(\Gamma, \delta) = (\Gamma_{d+1}, \delta_{d+1})$ is a reduced power circuit and for every node $P \in \Pi$ there exists a node $Q \in \Gamma_{d+1}$ such that $\varepsilon(Q) = \varepsilon(P)$ and M_{d+1} is a compact marking on Γ_{d+1} with $\varepsilon(M_{d+1}) = \varepsilon(M)$.

▷ **Claim 43** (see Claim 22). Let $d = \text{depth}(\Pi)$ and $\Gamma_0, \dots, \Gamma_{d+1}$ be as constructed above. Then for all i we have $|\mathcal{C}_{\Gamma_i}| \leq |\Pi| + 1$ and $|\Gamma_i| \leq (|\Pi| + 1)^2 \cdot (\log(|\Pi|) + 2)$.

Proof. According to Lemma 19 and Lemma 20 we have $|\mathcal{C}_{\Gamma_{i+1}}| \leq |\mathcal{C}_{\Gamma_i}| + |\text{Min}(\Xi_i)|$. Further observe that Π is the disjoint union of the $\text{Min}(\Xi_j)$ for $j \in [0..d]$. Since $|\mathcal{C}_{\Gamma_0}| = 1$, we obtain for all $i \in [0..d]$ that

$$|\mathcal{C}_{\Gamma_{i+1}}| \leq |\mathcal{C}_{\Gamma_i}| + |\text{Min}(\Xi_i)| \leq 1 + \sum_{0 \leq j \leq i} |\text{Min}(\Xi_j)| \leq |\Pi| + 1. \quad (3)$$

Again by Lemma 19 and Lemma 20 we have

$$\begin{aligned} |\Gamma_{i+1}| &\leq |\Gamma'_i| + |\mathcal{C}_{\Gamma'_i}| \cdot (\lceil \log(|\text{Min}(\Xi_i)|) \rceil + 1) && \text{(by Lemma 20)} \\ &\leq |\Gamma_i| + |\text{Min}(\Xi_i)| + (|\mathcal{C}_{\Gamma_i}| + |\text{Min}(\Xi_i)|) \cdot (\lceil \log(|\text{Min}(\Xi_i)|) \rceil + 1) && \text{(by Lemma 19)} \\ &\leq |\Gamma_i| + |\text{Min}(\Xi_i)| + (|\Pi| + 1) \cdot (\lceil \log(|\Pi|) \rceil + 1). && \text{(by (3))} \end{aligned}$$

Since $|\Gamma_0| = \lceil \log(|\Pi|) \rceil + 1$, we obtain by induction that

$$\begin{aligned} |\Gamma_i| &\leq |\Gamma_0| + \sum_{0 \leq j \leq i-1} |\text{Min}(\Xi_j)| + i \cdot (|\Pi| + 1) \cdot (\log(|\Pi|) + 2) \\ &\leq (\lceil \log(|\Pi|) \rceil + 1) + |\Pi| + i \cdot (|\Pi| + 1) \cdot (\log(|\Pi|) + 2) \\ &\leq (i + 1) \cdot (|\Pi| + 1) \cdot (\log(|\Pi|) + 2) \end{aligned}$$

for all $i \in [1..d + 1]$. The last inequality is due to the fact that $|\Pi| + 1 \geq 2$ and $\log(|\Pi|) + 2 \geq 2$. Since $d + 1 \leq |\Pi|$, we obtain $|\Gamma_i| \leq (|\Pi| + 1)^2 \cdot (\log(|\Pi|) + 2)$. ◁

74:24 Parallel Algorithms for the Baumslag Group

Let $D \in \mathbb{N}$ and assume that $\text{depth}(\Pi) \leq D$. By Lemma 19, Lemma 20 and Lemma 21 each iteration of the three steps above can be done in TC^0 . Notice here that the construction of the markings \tilde{M}_i and $\tilde{\Lambda}_P$ during `UPDATERMARKINGS` can be done in parallel – so it is in TC^0 , although Lemma 21 is stated only for a single marking. Now, the crucial observation is that, due to Claim 43, the input size for each iteration is polynomial in the original input size of (Π, δ_Π) . Therefore, we can compose the individual iterations and obtain a circuit of polynomial size and depth bounded by $\mathcal{O}(D)$. Thus, we have described a DepParaTC^0 circuit (parametrized by $\text{depth}(\Pi)$) for the problem of computing a reduced form for (Π, δ_Π) . This completes the proof of Theorem 17. ◀

The Complexity of Transitively Orienting Temporal Graphs

George B. Mertzios  

Department of Computer Science, Durham University, UK

Hendrik Molter  

Department of Industrial Engineering and Management, Ben-Gurion University of the Negev,
Beer Sheva, Israel

Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Malte Renken  

Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Paul G. Spirakis  

Department of Computer Science, University of Liverpool, UK

Computer Engineering & Informatics Department, University of Patras, Greece

Philipp Zschoche  

Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Abstract

In a *temporal network* with discrete time-labels on its edges, entities and information can only “flow” along sequences of edges whose time-labels are non-decreasing (resp. increasing), i.e. along temporal (resp. strict temporal) paths. Nevertheless, in the model for temporal networks of [Kempe, Kleinberg, Kumar, JCSS, 2002], the individual time-labeled edges remain undirected: an edge $e = \{u, v\}$ with time-label t specifies that “ u communicates with v at time t ”. This is a symmetric relation between u and v , and it can be interpreted that the information can flow in either direction. In this paper we make a first attempt to understand how the direction of information flow on one edge can impact the direction of information flow on other edges. More specifically, naturally extending the classical notion of a transitive orientation in static graphs, we introduce the fundamental notion of a *temporal transitive orientation* and we systematically investigate its algorithmic behavior in various situations. An orientation of a temporal graph is called *temporally transitive* if, whenever u has a directed edge towards v with time-label t_1 and v has a directed edge towards w with time-label $t_2 \geq t_1$, then u also has a directed edge towards w with some time-label $t_3 \geq t_2$. If we just demand that this implication holds whenever $t_2 > t_1$, the orientation is called *strictly temporally transitive*, as it is based on the fact that there is a strict directed temporal path from u to w . Our main result is a conceptually simple, yet technically quite involved, polynomial-time algorithm for recognizing whether a given temporal graph \mathcal{G} is transitively orientable. In wide contrast we prove that, surprisingly, it is NP-hard to recognize whether \mathcal{G} is strictly transitively orientable. Additionally we introduce and investigate further related problems to temporal transitivity, notably among them the *temporal transitive completion* problem, for which we prove both algorithmic and hardness results.

2012 ACM Subject Classification Theory of computation \rightarrow Graph algorithms analysis; Mathematics of computing \rightarrow Discrete mathematics

Keywords and phrases Temporal graph, transitive orientation, transitive closure, polynomial-time algorithm, NP-hardness, satisfiability

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.75

Related Version *Full Version:* <https://arxiv.org/abs/2102.06783> [36]

Funding *George B. Mertzios:* Supported by the EPSRC grant EP/P020372/1.

Hendrik Molter: Supported by the German Research Foundation (DFG), project MATE (NI 369/17), and by the Israeli Science Foundation (ISF), grant No. 1070/20.



© George B. Mertzios, Hendrik Molter, Malte Renken, Paul G. Spirakis, and Philipp Zschoche; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 75; pp. 75:1–75:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Malte Renken: Supported by the German Research Foundation (DFG), project MATE (NI 369/17).

Paul G. Spirakis: Supported by the NeST initiative of the School of EEE and CS at the University of Liverpool and by the EPSRC grant EP/P02002X/1.

1 Introduction

A *temporal* (or *dynamic*) network is, roughly speaking, a network whose underlying topology changes over time. This notion concerns a great variety of both modern and traditional networks; information and communication networks, social networks, and several physical systems are only few examples of networks which change over time [26,38,41]. Due to its vast applicability in many areas, the notion of temporal graphs has been studied from different perspectives under several different names such as *time-varying*, *evolving*, *dynamic*, and *graphs over time* (see [13–15] and the references therein). In this paper we adopt a simple and natural model for temporal networks which is given with discrete time-labels on the edges of a graph, while the vertex set remains unchanged. This formalism originates in the foundational work of Kempe et al. [27].

► **Definition 1** (Temporal Graph [27]). *A temporal graph is a pair $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$ is an underlying (static) graph and $\lambda : E \rightarrow \mathbb{N}$ is a time-labeling function which assigns to every edge of G a discrete-time label.*

Mainly motivated by the fact that, due to causality, entities and information in temporal graphs can only “flow” along sequences of edges whose time-labels are non-decreasing (resp. increasing), Kempe et al. introduced the notion of a (*strict*) *temporal path*, or (*strict*) *time-respecting path*, in a temporal graph (G, λ) as a path in G with edges e_1, e_2, \dots, e_k such that $\lambda(e_1) \leq \dots \leq \lambda(e_k)$ (resp. $\lambda(e_1) < \dots < \lambda(e_k)$). This notion of a temporal path naturally resembles the notion of a *directed* path in the classical static graphs, where the direction is from smaller to larger time-labels along the path. Nevertheless, in temporal paths the individual time-labeled edges remain undirected: an edge $e = \{u, v\}$ with time-label $\lambda(e) = t$ can be abstractly interpreted as “ u communicates with v at time t ”. Here the relation “communicates” is symmetric between u and v , i.e. it can be interpreted that the information can flow in either direction.

In this paper we make a first attempt to understand how the direction of information flow on one edge can impact the direction of information flow on other edges. More specifically, naturally extending the classical notion of a transitive orientation in static graphs [23], we introduce the fundamental notion of a *temporal transitive orientation* and we thoroughly investigate its algorithmic behavior in various situations. Imagine that v receives information from u at time t_1 , while w receives information from v at time $t_2 \geq t_1$. Then w *indirectly* receives information from u through the intermediate vertex v . Now, if the temporal graph correctly records the transitive closure of information passing, the directed edge from u to w must exist and must have a time label $t_3 \geq t_2$. In such a *transitively oriented* temporal graph, whenever an edge is oriented from a vertex u to a vertex w with time-label t , we have that *every* temporal path from u to w arrives no later than t , and that there is no temporal path from w to u . Different notions of temporal transitivity have also been used for automated temporal data mining [40] in medical applications [39], text processing [45]. Furthermore, in behavioral ecology, researchers have used a notion of orderly (transitive) triads A-B-C to quantify dominance among species. In particular, animal groups usually form dominance hierarchies in which dominance relations are transitive and can also change with time [32].

One natural motivation for our temporal transitivity notion may come from applications where confirmation and verification of information is vital, where vertices may represent entities such as investigative journalists or police detectives who gather sensitive information. Suppose that v queried some important information from u (the information source) at time t_1 , and afterwards, at time $t_2 \geq t_1$, w queried the important information from v (the intermediary). Then, in order to ensure the validity of the information received, w might want to verify it by *subsequently* querying the information directly from u at some time $t_3 \geq t_2$. Note that w might first receive the important information from u through various other intermediaries, and using several channels of different lengths. Then, to maximize confidence about the information, w should query u for verification only after receiving the information from the latest of these indirect channels.

It is worth noting here that the model of temporal graphs given in Definition 1 has been also used in its extended form, in which the temporal graph may contain multiple time-labels per edge [34]. This extended temporal graph model has been used to investigate temporal paths [3, 9, 11, 16, 34, 47] and other temporal path-related notions such as temporal analogues of distance and diameter [1], reachability [2] and exploration [1, 3, 20, 21], separation [22, 27, 48], and path-based centrality measures [12, 28], as well as recently non-path problems too such as temporal variations of coloring [37], vertex cover [4], matching [35], cluster editing [18], and maximal cliques [8, 25, 46]. However, in order to better investigate and illustrate the inherent combinatorial structure of temporal transitivity orientations, in this paper we mostly follow the original definition of temporal graphs given by Kempe et al. [27] with one time-label per edge [7, 17, 19]. Throughout the paper, whenever we assume multiple time-labels per edge we will state it explicitly; in all other cases we consider a single label per edge.

In static graphs, the transitive orientation problem has received extensive attention which resulted in numerous efficient algorithms. A graph is called *transitively orientable* (or a *comparability* graph) if it is possible to orient its edges such that, whenever we orient u towards v and v towards w , then the edge between u and w exists and is oriented towards w . The first polynomial-time algorithms for recognizing whether a given (static) graph G on n vertices and m edges is comparability (i.e. transitively orientable) were based on the notion of *forcing* an orientation and had running time $O(n^3)$ (see Golumbic [23] and the references therein). Faster algorithms for computing a transitive orientation of a given comparability graph have been later developed, having running times $O(n^2)$ [43] and $O(n + m \log n)$ [29], while the currently fastest algorithms run in linear $O(n + m)$ time and are based on efficiently computing a modular decomposition of G [30, 31]; see also Spinrad [44]. It is fascinating that, although all the latter algorithms compute a valid transitive orientation if G is a comparability graph, they fail to recognize whether the input graph is a comparability graph; instead they produce an orientation which is non-transitive if G is not a comparability graph. The fastest known algorithm for determining whether a given orientation is transitive requires matrix multiplication, currently achieved in $O(n^{2.37286})$ time [5].

Our contribution. In this paper we introduce the notion of *temporal transitive orientation* and we thoroughly investigate its algorithmic behavior in various situations. An orientation of a temporal graph $\mathcal{G} = (G, \lambda)$ is called *temporally transitive* if, whenever u has a directed edge towards v with time-label t_1 and v has a directed edge towards w with time-label $t_2 \geq t_1$, then u also has a directed edge towards w with some time-label $t_3 \geq t_2$. If we just demand that this implication holds whenever $t_2 > t_1$, the orientation is called *strictly* temporally transitive, as it is based on the fact that there is a strict directed temporal path from u to w . Similarly, if we demand that the transitive directed edge from u to w has time-label $t_3 > t_2$, the orientation is called *strongly* (resp. *strongly strictly*) temporally transitive.

Although these four natural variations of a temporally transitive orientation seem superficially similar to each other, it turns out that their computational complexity (and their underlying combinatorial structure) varies massively. Indeed we obtain a surprising result in Section 3: deciding whether a temporal graph \mathcal{G} admits a *temporally transitive* orientation is solvable in polynomial time (Section 3.2), while it is NP-hard to decide whether it admits a *strictly temporally transitive* orientation (Section 3.1). On the other hand, it turns out that, deciding whether \mathcal{G} admits a *strongly* or a *strongly strictly* temporal transitive orientation is (easily) solvable in polynomial time as they can both be reduced to 2SAT satisfiability.

Our main result is that, given a temporal graph $\mathcal{G} = (G, \lambda)$, we can decide in polynomial time whether \mathcal{G} is transitively orientable, and at the same time we can output a temporal transitive orientation if it exists. Although the analysis and correctness proof of our algorithm is technically quite involved, our algorithm is simple and easy to implement, as it is based on the notion of *forcing* an orientation.¹ Our algorithm extends and generalizes the classical polynomial-time algorithm for computing a transitive orientation in static graphs described by Golumbic [23]. The main technical difficulty in extending the algorithm from the static to the temporal setting is that, in temporal graphs we cannot simply use orientation forcings to eliminate the condition that a *triangle* is not allowed to be cyclically oriented. To resolve this issue, we first express the recognition problem of temporally transitively orientable graphs as a Boolean satisfiability problem of a *mixed* Boolean formula $\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$. Here $\phi_{3\text{NAE}}$ is a 3NAE (i.e. 3-NOT-ALL-EQUAL) formula and $\phi_{2\text{SAT}}$ is a 2SAT formula. Note that every clause $\text{NAE}(\ell_1, \ell_2, \ell_3)$ of $\phi_{3\text{NAE}}$ corresponds to the condition that a specific triangle in the temporal graph cannot be cyclically oriented. However, although deciding whether $\phi_{2\text{SAT}}$ is satisfiable can be done in linear time with respect to the size of the formula [6], the problem Not-All-Equal-3-SAT is NP-complete [42].

Our algorithm iteratively produces at iteration j a formula $\phi_{3\text{NAE}}^{(j)} \wedge \phi_{2\text{SAT}}^{(j)}$, which is computed from the previous formula $\phi_{3\text{NAE}}^{(j-1)} \wedge \phi_{2\text{SAT}}^{(j-1)}$ by (almost) simulating the classical greedy algorithm that solves 2SAT [6]. The 2SAT-algorithm proceeds greedily as follows. For every variable x_i , if setting $x_i = 1$ (resp. $x_i = 0$) leads to an immediate contradiction, the algorithm is forced to set $x_i = 0$ (resp. $x_i = 1$). Otherwise, if each of the truth assignments $x_i = 1$ and $x_i = 0$ does not lead to an immediate contradiction, the algorithm arbitrarily chooses to set $x_i = 1$ or $x_i = 0$, and thus some clauses are removed from the formula as they were satisfied. The argument for the correctness of the 2SAT-algorithm is that new clauses are *never added* to the formula at any step. The main technical difference between the 2SAT-algorithm and our algorithm is that, in our case, the formula $\phi_{3\text{NAE}}^{(j)} \wedge \phi_{2\text{SAT}}^{(j)}$ is *not* necessarily a sub-formula of $\phi_{3\text{NAE}}^{(j-1)} \wedge \phi_{2\text{SAT}}^{(j-1)}$, as in some cases we need to also add clauses. Our main technical result is that, nevertheless, at every iteration j the formula $\phi_{3\text{NAE}}^{(j)} \wedge \phi_{2\text{SAT}}^{(j)}$ is satisfiable if and only if $\phi_{3\text{NAE}}^{(j-1)} \wedge \phi_{2\text{SAT}}^{(j-1)}$ is satisfiable. The proof of this result (see Theorem 9) relies on a sequence of structural properties of temporal transitive orientations which we establish. This phenomenon of deducing a polynomial-time algorithm for an algorithmic graph problem by deciding satisfiability of a mixed Boolean formula (i.e. with both clauses of two and three literals) occurs rarely; this approach has been successfully used for the efficient recognition of simple-triangle (known also as “PI”) graphs [33].

In the second part of our paper (Section 4) we consider a natural extension of the temporal orientability problem, namely the *temporal transitive completion* problem. In this problem we are given a (partially oriented) temporal graph \mathcal{G} and a natural number k , and the question

¹ That is, orienting an edge from u to v *forces* us to orient another edge from a to b .

is whether it is possible to add at most k new edges (with the corresponding time-labels) to \mathcal{G} such that the resulting temporal graph is (strongly/strictly/strongly strictly) transitively orientable. We prove that all four versions of temporal transitive completion are NP-complete, even when the input temporal graph is completely unoriented. In contrast we show that, if the input temporal graph \mathcal{G} is *directed* (i.e. if every time-labeled edge has a fixed orientation) then all versions of temporal transitive completion are solvable in polynomial time. As a corollary of our results it follows that all four versions of temporal transitive completion are fixed-parameter-tractable (FPT) with respect to the number q of unoriented time-labeled edges in \mathcal{G} .

In the third and last part of our paper (Section 5) we consider the *multilayer transitive orientation* problem. In this problem we are given an undirected temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, and we ask whether there exists an orientation F of its edges (i.e. with exactly one orientation for each edge of G) such that, for every “time-layer” $t \geq 1$, the (static) oriented graph induced by the edges having time-label t is transitively oriented in F . Problem definitions of this type are commonly referred to as multilayer problems [10]. Observe that this problem trivially reduces to the static case if we assume that each edge has a single time-label, as then each layer can be treated independently of all others. However, if we allow \mathcal{G} to have multiple time-labels on every edge of G , then we show that the problem becomes NP-complete, even when every edge has at most two labels.

Due to space constraints, some of our results are deferred to a full version [36].

2 Preliminaries and Notation

Given a (static) undirected graph $G = (V, E)$, an edge between two vertices $u, v \in V$ is denoted by the unordered pair $\{u, v\} \in E$, and in this case the vertices u, v are said to be *adjacent*. If the graph is directed, we will use the ordered pair (u, v) (resp. (v, u)) to denote the oriented edge from u to v (resp. from v to u). For simplicity of the notation, we will usually drop the parentheses and the comma when denoting an oriented edge, i.e. we will denote (u, v) just by uv . Furthermore, $\widehat{uv} = \{uv, vu\}$ is used to denote the set of both oriented edges uv and vu between the vertices u and v .

Let $S \subseteq E$ be a subset of the edges of an undirected (static) graph $G = (V, E)$, and let $\widehat{S} = \{uv, vu : \{u, v\} \in S\}$ be the set of both possible orientations uv and vu of every edge $\{u, v\} \in S$. Let $F \subseteq \widehat{S}$. If F contains *at least one* of the two possible orientations uv and vu of each edge $\{u, v\} \in S$, then F is called an *orientation* of the edges of S . F is called a *proper orientation* if it contains *exactly one* of the orientations uv and vu of every edge $\{u, v\} \in S$. Note here that, in order to simplify some technical proofs, the above definition of an orientation allows F to be not proper, i.e. to contain *both* uv and vu for a specific edge $\{u, v\}$. However, whenever F is not proper, this means that F can be discarded as it cannot be used as a part of a (temporal) transitive orientation. For every orientation F denote by $F^{-1} = \{vu : uv \in F\}$ the *reversal* of F . Note that $F \cap F^{-1} = \emptyset$ if and only if F is proper.

In a temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, whenever $\lambda(\{v, w\}) = t$ (or simply $\lambda(v, w) = t$), we refer to the tuple $(\{v, w\}, t)$ as a *time-edge* of \mathcal{G} . A triangle of (G, λ) on the vertices u, v, w is a *synchronous triangle* if $\lambda(u, v) = \lambda(v, w) = \lambda(w, u)$. Let $G = (V, E)$ and let F be a proper orientation of the whole edge set E . Then (\mathcal{G}, F) , or (G, λ, F) , is a *proper orientation* of the temporal graph \mathcal{G} . A *partial proper orientation* F of $\mathcal{G} = (G, \lambda)$ is an orientation of a subset of E . To indicate that the edge $\{u, v\}$ of a time-edge $(\{u, v\}, t)$ is oriented from u to v (that is, $uv \in F$ in a (partial) proper orientation F), we use the term $((u, v), t)$, or simply (uv, t) . For simplicity we may refer to a (partial) proper orientation just as a (partial) orientation, whenever the term “proper” is clear from the context.

75:6 The Complexity of Transitively Orienting Temporal Graphs

A static graph $G = (V, E)$ is a *comparability graph* if there exists a proper orientation F of E which is *transitive*, that is, if $F \cap F^{-1} = \emptyset$ and $F^2 \subseteq F$, where $F^2 = \{uv : uv, vw \in F\}$ [23]. Analogously, in a temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, we define a proper orientation F of E to be *temporally transitive*, if:

whenever (uv, t_1) and (vw, t_2) are oriented time-edges in (\mathcal{G}, F) such that $t_2 \geq t_1$, there exists an oriented time-edge (wu, t_3) in (\mathcal{G}, F) , for some $t_3 \geq t_2$.

In the above definition of a temporally transitive orientation, if we replace the condition “ $t_3 \geq t_2$ ” with “ $t_3 > t_2$ ”, then F is called *strongly temporally transitive*. If we instead replace the condition “ $t_2 \geq t_1$ ” with “ $t_2 > t_1$ ”, then F is called *strictly temporally transitive*. If we do both of these replacements, then F is called *strongly strictly temporally transitive*. Note that strong (strict) temporal transitivity implies (strict) temporal transitivity, while (strong) temporal transitivity implies (strong) strict temporal transitivity. Furthermore, similarly to the established terminology for static graphs, we define a temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, to be a (*strongly/strictly*) *temporal comparability graph* if there exists a proper orientation F of E which is (*strongly/strictly*) *temporally transitive*.

We are now ready to formally introduce the following decision problem of recognizing whether a given temporal graph is temporally transitively orientable or not.

TEMPORAL TRANSITIVE ORIENTATION (TTO)

Input: A temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$.

Question: Does \mathcal{G} admit a temporally transitive orientation F of E ?

In the above problem definition of TTO, if we ask for the existence of a strictly (resp. strongly, or strongly strictly) temporally transitive orientation F , we obtain the decision problem STRICT (resp. STRONG, or STRONG STRICT) TEMPORAL TRANSITIVE ORIENTATION (TTO).

Let $\mathcal{G} = (G, \lambda)$ be a temporal graph, where $G = (V, E)$. Let $G' = (V, E')$ be a graph such that $E \subseteq E'$, and let $\lambda' : E' \rightarrow \mathbb{N}$ be a time-labeling function such that $\lambda'(u, v) = \lambda(u, v)$ for every $\{u, v\} \in E$. Then the temporal graph $\mathcal{G}' = (G', \lambda')$ is called a *temporal supergraph* of \mathcal{G} . We can now define our next problem definition regarding computing temporally orientable supergraphs of \mathcal{G} .

TEMPORAL TRANSITIVE COMPLETION (TTC)

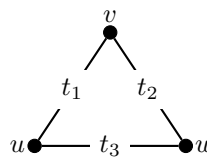
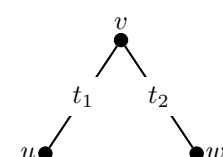
Input: A temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$, a (partial) orientation F of \mathcal{G} , and an integer k .

Question: Does there exist a temporal supergraph $\mathcal{G}' = (G', \lambda')$ of (G, λ) , where $G' = (V, E')$, and a transitive orientation $F' \supseteq F$ of \mathcal{G}' such that $|E' \setminus E| \leq k$?

Similarly to TTO, if we ask in the problem definition of TTC for the existence of a strictly (resp. strongly, or strongly strictly) temporally transitive orientation F' , we obtain the decision problem STRICT (resp. STRONG, or STRONG STRICT) TEMPORAL TRANSITIVE COMPLETION (TTC).

Now we define our final problem which asks for an orientation F of a temporal graph $\mathcal{G} = (G, \lambda)$ (i.e. with exactly one orientation for each edge of G) such that, for every “time-layer” $t \geq 1$, the (static) oriented graph defined by the edges having time-label t is transitively oriented in F . This problem does not make much sense if every edge has exactly one time-label in \mathcal{G} , as in this case it can be easily solved by just repeatedly applying any known static transitive orientation algorithm. Therefore, in the next problem definition, we assume that in the input temporal graph $\mathcal{G} = (G, \lambda)$ every edge of G potentially has multiple time-labels, i.e. the time-labeling function is $\lambda : E \rightarrow 2^{\mathbb{N}}$.

■ **Table 1** Orientation conditions imposed by a triangle (left) and an induced path of length two (right) in the underlying graph G for the decision problems (STRICT/STRONG/STRONG STRICT) TTO. Here, \top means that no restriction is imposed, \perp means that the graph is not orientable, and in the case of triangles, “non-cyclic” means that all orientations except the ones that orient the triangle cyclicly are allowed.

					
	$t_1 = t_2 = t_3$	$t_1 < t_2 = t_3$	$t_1 \leq t_2 < t_3$	$t_1 = t_2$	$t_1 < t_2$
TTO	non-cyclic	$wu = wv$	$vw \implies uw$ $vu \implies wu$	$uv = wv$	$uv \implies vw$
STRONG TTO	\perp	$wu \wedge wv$	$vw \implies uw$ $vu \implies wu$	$uv = wv$	$uv \implies vw$
STRICT TTO	\top	non-cyclic	$vw \implies uw$ $vu \implies wu$	\top	$uv \implies vw$
STR. STR. TTO	\top	$vu \implies wu$ $uv \implies wv$	$vw \implies uw$ $vu \implies wu$	\top	$uv \implies vw$

MULTILAYER TRANSITIVE ORIENTATION (MTO)

Input: A temporal graph $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$ and $\lambda : E \rightarrow 2^{\mathbb{N}}$.

Question: Is there an orientation F of the edges of G such that, for every $t \geq 1$, the (static) oriented graph induced by the edges having time-label t is transitively oriented?

3 The recognition of temporally transitively orientable graphs

In this section we investigate the computational complexity of all variants of TTO. We show that TTO as well as the two variants STRONG TTO and STRONG STRICT TTO, are solvable in polynomial time, whereas STRICT TTO turns out to be NP-complete.

The main idea of our approach to solve TTO and its variants is to create Boolean variables for each edge of the underlying graph G and interpret setting a variable to 1 or 0 with the two possible ways of directing the corresponding edge.

More formally, for every edge $\{u, v\}$ we introduce a variable x_{uv} and setting this variable to 1 corresponds to the orientation uv while setting this variable to 0 corresponds to the orientation vu . Now consider the example of Figure 1(a), i.e. an induced path of length two in the underlying graph G on three vertices u, v, w , and let $\lambda(u, v) = 1$ and $\lambda(v, w) = 2$. Then the orientation uv “forces” the orientation wv . Indeed, if we otherwise orient $\{v, w\}$ as vw , then the edge $\{u, w\}$ must exist and be oriented as uw in any temporal transitive orientation, which is a contradiction as there is no edge between u and w . We can express this “forcing” with the implication $x_{uv} \implies x_{wv}$. In this way we can deduce the constraints that all triangles or induced paths on three vertices impose on any (strong/strict/strong strict) temporal transitive orientation. We collect all these constraints in Table 1.

When looking at the conditions imposed on temporal transitive orientations collected in Table 1, we can observe that all conditions except “non-cyclic” are expressible in 2SAT. Since 2SAT is solvable in linear time [6], it immediately follows that the strong variants of temporal transitivity are solvable in polynomial time, as the next theorem states.

► **Theorem 2.** *STRONG TTO and STRONG STRICT TTO are solvable in polynomial time.*

In the variants TTO and STRICT TTO, however, we can have triangles which impose a “non-cyclic” orientation of three edges (Table 1). This can be naturally modeled by a not-all-equal (NAE) clause.² However, if we now naïvely model the conditions with a Boolean formula, we obtain a formula with 2SAT clauses and 3NAE clauses. Deciding whether such a formula is satisfiable is NP-complete in general [42]. Hence, we have to investigate these two variants more thoroughly.

The only difference between the triangles that impose these “non-cyclic” orientations in these two problem variants is that, in TTO, the triangle is *synchronous* (i.e. all its three edges have the same time-label), while in STRICT TTO two of the edges are synchronous and the third one has a smaller time-label than the other two. As it turns out, this difference of the two problem variants has important implications on their computational complexity. In fact, we obtain a surprising result: TTO is solvable in polynomial time while STRICT TTO is NP-complete.

3.1 Strict TTO is NP-Complete

In this section we show that in contrast to the other variants, STRICT TTO is NP-complete.

► **Theorem 3.** *STRICT TTO is NP-complete even if the temporal input graph has only four different time labels.*

3.2 A polynomial-time algorithm for TTO

Let $G = (V, E)$ be a static undirected graph. There are various polynomial-time algorithms for deciding whether G admits a transitive orientation F . However our results in this section are inspired by the transitive orientation algorithm described by Golumbic [23], which is based on the crucial notion of *forcing* an orientation. The notion of forcing in static graphs is illustrated in Figure 1 (a): if we orient the edge $\{u, v\}$ as uv (i.e., from u to v) then we are forced to orient the edge $\{v, w\}$ as wv (i.e., from w to v) in any transitive orientation F of G . Indeed, if we otherwise orient $\{v, w\}$ as vw (i.e. from v to w), then the edge $\{u, w\}$ must exist and it must be oriented as uw in any transitive orientation F of G , which is a contradiction as $\{u, w\}$ is not an edge of G . Similarly, if we orient the edge $\{u, v\}$ as vu then we are forced to orient the edge $\{v, w\}$ as vw . That is, in any transitive orientation F of G we have that $uv \in F \Leftrightarrow vw \in F$. This forcing operation can be captured by the binary forcing relation Γ which is defined on the edges of a static graph G as follows [23].

$$uv \Gamma u'v' \quad \text{if and only if} \quad \begin{cases} \text{either } u = u' \text{ and } \{v, v'\} \notin E \\ \text{or } v = v' \text{ and } \{u, u'\} \notin E \end{cases} . \quad (1)$$

We now extend the definition of Γ in a natural way to the binary relation Λ on the edges of a temporal graph (G, λ) , see Equation (2). For this, observe from Table 1 that the only cases, where we have $uv \in F \Leftrightarrow vw \in F$ in any temporal transitive orientation of (G, λ) , are

² A not all equal clause is a set of literals and it evaluates to **true** if and only if at least two literals in the set evaluate to different truth values.



■ **Figure 1** The orientation uv forces the orientation wu and vice-versa in the examples of (a) a static graph G where $\{u, v\}, \{v, w\} \in E(G)$ and $\{u, w\} \notin E(G)$, and of (b) a temporal graph (G, λ) where $\lambda(u, w) = 3 < 5 = \lambda(u, v) = \lambda(v, w)$.

when (i) the vertices u, v, w induce a path of length 2 (see Figure 1 (a)) and $\lambda(u, v) = \lambda(v, w)$, as well as when (ii) u, v, w induce a triangle and $\lambda(u, w) < \lambda(u, v) = \lambda(v, w)$. The latter situation is illustrated in the example of Figure 1 (b). The binary forcing relation Λ is only defined on pairs of edges $\{u, v\}$ and $\{u', v'\}$ where $\lambda(u, v) = \lambda(u', v')$, as follows.

$$uv \Lambda u'v' \text{ if and only if } \lambda(u, v) = \lambda(u', v') = t \text{ and } \begin{cases} u = u' \text{ and } \{v, v'\} \notin E, \text{ or} \\ v = v' \text{ and } \{u, u'\} \notin E, \text{ or} \\ u = u' \text{ and } \lambda(v, v') < t, \text{ or} \\ v = v' \text{ and } \lambda(u, u') < t. \end{cases} \quad (2)$$

Note that, for every edge $\{u, v\} \in E$ we have that $uv \Lambda uv$. The forcing relation Λ for temporal graphs shares some properties with the forcing relation Γ for static graphs. In particular, the reflexive transitive closure Λ^* of Λ is an equivalence relation, which partitions the edges of each set $E_t = \{\{u, v\} \in E : \lambda(u, v) = t\}$ into its Λ -implication classes (or simply, into its *implication classes*). Two edges $\{a, b\}$ and $\{c, d\}$ are in the same Λ -implication class if and only $ab \Lambda^* cd$, i.e. there exists a sequence $ab = a_0b_0 \Lambda a_1b_1 \Lambda \dots \Lambda a_kb_k = cd$, with $k \geq 0$. Note that, for this to happen, we must have $\lambda(a_0, b_0) = \lambda(a_1, b_1) = \dots = \lambda(a_k, b_k) = t$ for some $t \geq 1$. Such a sequence is called a Λ -chain from ab to cd , and we say that ab (eventually) Λ -forces cd . Furthermore note that $ab \Lambda^* cd$ if and only if $ba \Lambda^* dc$. For the next lemma, we use the notation $\widehat{A} = \{uv, vu : uv \in A\}$.

► **Lemma 4.** *Let A be a Λ -implication class of a temporal graph (G, λ) . Then either $A = A^{-1} = \widehat{A}$ or $A \cap A^{-1} = \emptyset$.*

► **Definition 5.** *Let F be a proper orientation and A be a Λ -implication class of a temporal graph (G, λ) . If $A \subseteq F$, we say that F respects A .*

► **Lemma 6.** *Let F be a proper orientation and A be a Λ -implication class of a temporal graph (G, λ) . Then F respects either A or A^{-1} (i.e. either $A \subseteq F$ or $A^{-1} \subseteq F$), and in either case $A \cap A^{-1} = \emptyset$.*

The next lemma, which is crucial for proving the correctness of our algorithm, extends an important known property of the forcing relation Γ for static graphs [23, Lemma 5.3] to the temporal case.

► **Lemma 7 (Temporal Triangle Lemma).** *Let (G, λ) be a temporal graph and with a synchronous triangle on the vertices a, b, c , where $\lambda(a, b) = \lambda(b, c) = \lambda(c, a) = t$. Let A, B, C be three Λ -implication classes of (G, λ) , where $ab \in C$, $bc \in A$, and $ca \in B$, where $A \neq B^{-1}$ and $A \neq C^{-1}$.*

1. *If some $b'c' \in A$, then $ab' \in C$ and $c'a \in B$.*
2. *If some $b'c' \in A$ and $a'b' \in C$, then $c'a' \in B$.*
3. *No edge of A touches vertex a .*

Deciding temporal transitivity using Booleansatisfiability. Starting with any undirected edge $\{u, v\}$ of the underlying graph G , we can clearly enumerate in polynomial time the whole Λ -implication class A to which the oriented edge uv belongs (cf. Equation (2)). If the reversely directed edge $vu \in A$ then Lemma 4 implies that $A = A^{-1} = \widehat{A}$. Otherwise, if $vu \notin A$ then $vu \in A^{-1}$ and Lemma 4 implies that $A \cap A^{-1} = \emptyset$. Thus, we can also decide in polynomial time whether $A \cap A^{-1} = \emptyset$. If we encounter a Λ -implication class A such that $A \cap A^{-1} \neq \emptyset$, then it follows by Lemma 6 that (G, λ) is not temporally transitively orientable.

In the remainder of the section we will assume that $A \cap A^{-1} = \emptyset$ for every Λ -implication class A of (G, λ) , which is a *necessary* condition for (G, λ) to be temporally transitively orientable. Moreover it follows by Lemma 6 that, if (G, λ) admits a temporally transitively orientation F , then either $A \subseteq F$ or $A^{-1} \subseteq F$. This allows us to define a Boolean variable x_A for every Λ -implication class A , where $x_A = \overline{x_{A^{-1}}}$. Here $x_A = 1$ (resp. $x_{A^{-1}} = 1$) means that $A \subseteq F$ (resp. $A^{-1} \subseteq F$), where F is the temporally transitively orientation which we are looking for. Let $\{A_1, A_2, \dots, A_s\}$ be a set of Λ -implication classes such that $\{\widehat{A}_1, \widehat{A}_2, \dots, \widehat{A}_s\}$ is a partition of the edges of the underlying graph G .³ Then any truth assignment τ of the variables x_1, x_2, \dots, x_s (where $x_i = x_{A_i}$ for every $i = 1, 2, \dots, s$) corresponds bijectively to one possible orientation of the temporal graph (G, λ) , in which every Λ -implication class is oriented consistently.

Now we define two Boolean formulas $\phi_{3\text{NAE}}$ and $\phi_{2\text{SAT}}$ such that (G, λ) admits a temporal transitively orientation if and only if there is a truth assignment τ of the variables x_1, x_2, \dots, x_s such that both $\phi_{3\text{NAE}}$ and $\phi_{2\text{SAT}}$ are simultaneously satisfied. Intuitively, $\phi_{3\text{NAE}}$ captures the “non-cyclic” condition from Table 1 while $\phi_{2\text{SAT}}$ captures the remaining conditions. Here $\phi_{3\text{NAE}}$ is a 3NAE formula, i.e., the disjunction of clauses with three literals each, where every clause $\text{NAE}(\ell_1, \ell_2, \ell_3)$ is satisfied if and only if at least one of the literals $\{\ell_1, \ell_2, \ell_3\}$ is equal to 1 and at least one of them is equal to 0. Furthermore $\phi_{2\text{SAT}}$ is a 2SAT formula, i.e., the disjunction of 2CNF clauses with two literals each, where every clause $(\ell_1 \vee \ell_2)$ is satisfied if and only if at least one of the literals $\{\ell_1, \ell_2\}$ is equal to 1.

For simplicity of the presentation we also define a variable x_{uv} for every directed edge uv . More specifically, if $uv \in A_i$ (resp. $uv \in A_i^{-1}$) then we set $x_{uv} = x_i$ (resp. $x_{uv} = \overline{x_i}$). That is, $x_{uv} = \overline{x_{vu}}$ for every undirected edge $\{u, v\} \in E$. Note that, although $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$ are defined as variables, they can equivalently be seen as *literals* in a Boolean formula over the variables x_1, x_2, \dots, x_s . The process of building all Λ -implication classes and all variables $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$ is given by Algorithm 1.

Description of the 3NAE formula $\phi_{3\text{NAE}}$. The formula $\phi_{3\text{NAE}}$ captures the “non-cyclic” condition of the problem variant TTO (presented in Table 1). The formal description of $\phi_{3\text{NAE}}$ is as follows. Consider a synchronous triangle of (G, λ) on the vertices u, v, w . Assume that $x_{uv} = x_{vw}$, i.e., x_{uv} is the same variable as x_{vw} . Then the pair $\{uv, vw\}$ of oriented edges belongs to the same Λ -implication class A_i . This implies that the triangle on the vertices u, v, w is never cyclically oriented in any proper orientation F that respects A_i or A_i^{-1} . Note that, by symmetry, the same happens if $x_{vw} = x_{uw}$ or if $x_{wu} = x_{vu}$. Assume, on the contrary, that $x_{uv} \neq x_{vw}$, $x_{vw} \neq x_{uw}$, and $x_{wu} \neq x_{vu}$. In this case we add to $\phi_{3\text{NAE}}$ the clause $\text{NAE}(x_{uv}, x_{vw}, x_{wu})$. Note that the triangle on u, v, w is transitively oriented if and only if $\text{NAE}(x_{uv}, x_{vw}, x_{wu})$ is satisfied, i.e., at least one of the variables $\{x_{uv}, x_{vw}, x_{wu}\}$ receives the value 1 and at least one of them receives the value 0.

³ Here we slightly abuse the notation by identifying the undirected edge $\{u, v\}$ with the set of both its orientations $\{uv, vu\}$.

■ **Algorithm 1** Building the Λ -implication classes and the edge-variables.

Input: A temporal graph (G, λ) , where $G = (V, E)$.

Output: The variables $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$, or the announcement that (G, λ) is temporally not transitively orientable.

```

1:  $s \leftarrow 0$ ;  $E_0 \leftarrow E$ 
2: while  $E_0 \neq \emptyset$  do
3:    $s \leftarrow s + 1$ ; Let  $\{p, q\} \in E_0$  be arbitrary
4:   Build the  $\Lambda$ -implication class  $A_s$  of the oriented edge  $pq$  (by Equation (2))
5:   if  $qp \in A_s$  then  $\{A_s \cap A_s^{-1} \neq \emptyset\}$ 
6:     return “NO”
7:   else
8:      $x_s$  is the variable corresponding to the directed edges of  $A_s$ 
9:     for every  $uv \in A_s$  do
10:       $x_{uv} \leftarrow x_s$ ;  $x_{vu} \leftarrow \overline{x_s}$   $\{x_{uv}$  and  $x_{vu}$  become aliases of  $x_s$  and  $\overline{x_s}\}$ 
11:     $E_0 \leftarrow E_0 \setminus \widehat{A_s}$ 
12: return  $\Lambda$ -implication classes  $\{A_1, A_2, \dots, A_s\}$  and variables  $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$ 

```

Description of the 2SAT formula $\phi_{2\text{SAT}}$. The formula $\phi_{2\text{SAT}}$ captures all conditions apart from the “non-cyclic” condition of the problem variant TTO (presented in Table 1). The formal description of $\phi_{2\text{SAT}}$ is as follows. Consider a triangle of (G, λ) on the vertices u, v, w , where $\lambda(u, v) = t_1$, $\lambda(v, w) = t_2$, $\lambda(w, v) = t_3$, and $t_1 \leq t_2 \leq t_3$. If $t_1 < t_2 = t_3$ then we add to $\phi_{2\text{SAT}}$ the clauses $(x_{uv} \vee x_{vw}) \wedge (x_{vw} \vee x_{wu})$; note that these clauses are equivalent to $x_{wu} = x_{vw}$. If $t_1 \leq t_2 < t_3$ then we add to $\phi_{2\text{SAT}}$ the clauses $(x_{vw} \vee x_{uv}) \wedge (x_{uv} \vee x_{wu})$; note that these clauses are equivalent to $(x_{vw} \Rightarrow x_{uv}) \wedge (x_{vu} \Rightarrow x_{wu})$. Now consider a path of length 2 that is induced by the vertices u, v, w , where $\lambda(u, v) = t_1$, $\lambda(v, w) = t_2$, and $t_1 \leq t_2$. If $t_1 = t_2$ then we add to $\phi_{2\text{SAT}}$ the clauses $(x_{vu} \vee x_{wv}) \wedge (x_{vw} \vee x_{uv})$; note that these clauses are equivalent to $(x_{uv} = x_{wv})$. Finally, if $t_1 < t_2$ then we add to $\phi_{2\text{SAT}}$ the clause $(x_{vu} \vee x_{wv})$; note that this clause is equivalent to $(x_{uv} \Rightarrow x_{wv})$.

Brief outline of the algorithm. In the *initialization phase*, we exhaustively check which truth values are *forced* in $\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$ by using the subroutine INITIAL-FORCING. During the execution of INITIAL-FORCING, we either replace the formulas $\phi_{3\text{NAE}}$ and $\phi_{2\text{SAT}}$ by the equivalent formulas $\phi_{3\text{NAE}}^{(0)}$ and $\phi_{2\text{SAT}}^{(0)}$, respectively, or we reach a contradiction by showing that $\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$ is unsatisfiable.

► **Observation 8.** *The temporal graph (G, λ) is transitively orientable if and only if $\phi_{3\text{NAE}}^{(0)} \wedge \phi_{2\text{SAT}}^{(0)}$ is satisfiable.*

The *main phase* of the algorithm starts once the formulas $\phi_{3\text{NAE}}^{(0)}$ and $\phi_{2\text{SAT}}^{(0)}$ have been computed. Then we iteratively try assigning to each variable x_i the truth value 1 or 0. Once we have set $x_i = 1$ (resp. $x_i = 0$) during the iteration $j \geq 1$ of the algorithm, we call algorithm BOOLEAN-FORCING (see Algorithm 3) as a subroutine to check which implications this value of x_i has on the current formulas $\phi_{3\text{NAE}}^{(j-1)}$ and $\phi_{2\text{SAT}}^{(j-1)}$ and which other truth values of variables are forced. The correctness of BOOLEAN-FORCING can be easily verified by checking all subcases of BOOLEAN-FORCING. During the execution of BOOLEAN-FORCING, we either replace the current formulas by $\phi_{3\text{NAE}}^{(j)}$ and $\phi_{2\text{SAT}}^{(j)}$, or we reach a contradiction by showing that, setting $x_i = 1$ (resp. $x_i = 0$) makes $\phi_{3\text{NAE}}^{(j-1)} \wedge \phi_{2\text{SAT}}^{(j-1)}$ unsatisfiable. If each of the truth assignments $\{x_i = 1, x_i = 0\}$ leads to such a contradiction, we return that (G, λ)

75:12 The Complexity of Transitively Orienting Temporal Graphs

■ **Algorithm 2** INITIAL-FORCING.

Input: A 2-SAT formula $\phi_{2\text{SAT}}$ and a 3-NAE formula $\phi_{3\text{NAE}}$

Output: A 2-SAT formula $\phi_{2\text{SAT}}^{(0)}$ and a 3-NAE formula $\phi_{3\text{NAE}}^{(0)}$ such that $\phi_{2\text{SAT}}^{(0)} \wedge \phi_{3\text{NAE}}^{(0)}$ is satisfiable if and only if $\phi_{2\text{SAT}} \wedge \phi_{3\text{NAE}}$ is satisfiable, or the announcement that $\phi_{2\text{SAT}} \wedge \phi_{3\text{NAE}}$ is not satisfiable.

- 1: $\phi_{3\text{NAE}}^{(0)} \leftarrow \phi_{3\text{NAE}}$; $\phi_{2\text{SAT}}^{(0)} \leftarrow \phi_{2\text{SAT}}$ {initialization}
- 2: **for** every variable x_i appearing in $\phi_{3\text{NAE}}^{(0)} \wedge \phi_{2\text{SAT}}^{(0)}$ **do**
- 3: **if** BOOLEAN-FORCING $\left(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}, x_i, 1\right) = \text{“NO”}$ **then**
- 4: **if** BOOLEAN-FORCING $\left(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}, x_i, 0\right) = \text{“NO”}$ **then**
- 5: **return** “NO” {both $x_i = 1$ and $x_i = 0$ invalidate the formulas}
- 6: **else**
- 7: $\left(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}\right) \leftarrow \text{BOOLEAN-FORCING}\left(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}, x_i, 0\right)$
- 8: **else**
- 9: **if** BOOLEAN-FORCING $\left(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}, x_i, 0\right) = \text{“NO”}$ **then**
- 10: $\left(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}\right) \leftarrow \text{BOOLEAN-FORCING}\left(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}, x_i, 1\right)$
- 11: **for** every clause NAE(x_{uv}, x_{vw}, x_{wu}) of $\phi_{3\text{NAE}}^{(0)}$ **do**
- 12: **for** every variable x_{ab} **do**
- 13: **if** $x_{ab} \xrightarrow{*}_{\phi_{2\text{SAT}}^{(0)}} x_{uv}$ and $x_{ab} \xrightarrow{*}_{\phi_{2\text{SAT}}^{(0)}} x_{vw}$ **then** {add $(x_{ab} \Rightarrow x_{uw})$ to $\phi_{2\text{SAT}}^{(0)}$ }
- 14: $\phi_{2\text{SAT}}^{(0)} \leftarrow \phi_{2\text{SAT}}^{(0)} \wedge (x_{ba} \vee x_{uw})$
- 15: Repeat lines 2 and 11 until no changes occur on $\phi_{2\text{SAT}}^{(0)}$ and $\phi_{3\text{NAE}}^{(0)}$
- 16: **return** $\left(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}\right)$

is a *no*-instance. Otherwise, if at least one of the truth assignments $\{x_i = 1, x_i = 0\}$ does not lead to such a contradiction, we follow this truth assignment and proceed with the next variable.

As we prove in our *main technical result* of this section (Theorem 9), $\phi_{3\text{NAE}}^{(j-1)} \wedge \phi_{2\text{SAT}}^{(j-1)}$ is satisfiable if and only if $\phi_{3\text{NAE}}^{(j)} \wedge \phi_{2\text{SAT}}^{(j)}$ is satisfiable. Note that, during the execution of the algorithm, we can *both add and remove* clauses from $\phi_{2\text{SAT}}^{(j)}$. On the other hand, we can *only remove* clauses from $\phi_{3\text{NAE}}^{(j)}$. Thus, at some iteration j , we obtain $\phi_{3\text{NAE}}^{(j)} = \emptyset$, and after that iteration we only need to decide satisfiability of $\phi_{2\text{SAT}}^{(j)}$ which can be done efficiently [6].

We are now ready to present in the next theorem our main technical result of this section.

► **Theorem 9.** *For every iteration $j \geq 1$ of the algorithm, $\phi_{3\text{NAE}}^{(j)} \wedge \phi_{2\text{SAT}}^{(j)}$ is satisfiable if and only if $\phi_{3\text{NAE}}^{(j-1)} \wedge \phi_{2\text{SAT}}^{(j-1)}$ is satisfiable.*

Using Theorem 9, we can now conclude this section with the next theorem.

► **Theorem 10.** *TTO can be solved in polynomial time.*

Proof sketch. First recall by Observation 8 that the input temporal graph (G, λ) is transitively orientable if and only if $\phi_{3\text{NAE}}^{(0)} \wedge \phi_{2\text{SAT}}^{(0)}$ is satisfiable.

■ **Algorithm 3** BOOLEAN-FORCING.

Input: A 2-SAT formula ϕ_2 , a 3-NAE formula ϕ_3 , and a variable x_i of $\phi_2 \wedge \phi_3$, and a truth value $\text{VALUE} \in \{0, 1\}$

Output: A 2-SAT formula ϕ'_2 and a 3-NAE formula ϕ'_3 , obtained from ϕ_2 and ϕ_3 by setting $x_i = \text{VALUE}$, or the announcement that $x_i = \text{VALUE}$ does not satisfy $\phi_2 \wedge \phi_3$.

```

1:  $\phi'_2 \leftarrow \phi_2$ ;  $\phi'_3 \leftarrow \phi_3$ 
2: while  $\phi'_2$  has a clause  $(x_{uv} \vee x_{pq})$  and  $x_{uv} = 1$  do
3:   Remove the clause  $(x_{uv} \vee x_{pq})$  from  $\phi'_2$ 
4: while  $\phi'_2$  has a clause  $(x_{uv} \vee x_{pq})$  and  $x_{uv} = 0$  do
5:   if  $x_{pq} = 0$  then return “NO”
6:   Remove the clause  $(x_{uv} \vee x_{pq})$  from  $\phi'_2$ ;  $x_{pq} \leftarrow 1$ 
7: for every variable  $x_{uv}$  that does not yet have a truth value do
8:   if  $x_{uv} \xrightarrow{*} \phi'_2 x_{vu}$ , where  $\phi''_2 = \phi'_2 \setminus \phi_2$  then  $x_{uv} \leftarrow 0$ 
9:   for every clause  $\text{NAE}(x_{uv}, x_{vw}, x_{wu})$  of  $\phi'_3$  do {synchronous triangle on vertices  $u, v, w$ }
10:  if  $x_{uv} \xrightarrow{*} \phi'_2 x_{vw}$  then {add  $(x_{uv} \Rightarrow x_{uw}) \wedge (x_{uv} \Rightarrow x_{vw})$  to  $\phi'_2$ }
11:   $\phi'_2 \leftarrow \phi'_2 \wedge (x_{vu} \vee x_{uw}) \wedge (x_{wu} \vee x_{vw})$ 
12:  Remove the clause  $\text{NAE}(x_{uv}, x_{vw}, x_{wu})$  from  $\phi'_3$ 
13:  if  $x_{uv}$  already got the value 1 or 0 then
14:    Remove the clause  $\text{NAE}(x_{uv}, x_{vw}, x_{wu})$  from  $\phi'_3$ 
15:    if  $x_{vw}$  and  $x_{wu}$  do not have yet a truth value then
16:      if  $x_{uv} = 1$  then {add  $(x_{vw} \Rightarrow x_{uw})$  to  $\phi'_2$ }
17:       $\phi'_2 \leftarrow \phi'_2 \wedge (x_{vw} \vee x_{uw})$ 
18:      else { $x_{uv} = 0$ ; in this case add  $(x_{uv} \Rightarrow x_{vw})$  to  $\phi'_2$ }
19:       $\phi'_2 \leftarrow \phi'_2 \wedge (x_{wu} \vee x_{vw})$ 
20:      if  $x_{vw} = x_{uv}$  and  $x_{wu}$  does not have yet a truth value then
21:         $x_{wu} \leftarrow 1 - x_{uv}$ 
22:      if  $x_{vw} = x_{wu} = x_{uv}$  then return “NO”
23: Repeat lines 2, 4, 7, and 9 until no changes occur on  $\phi'_2$  and  $\phi'_3$ 
24: if both  $x_{uv} = 0$  and  $x_{uv} = 1$  for some variable  $x_{uv}$  then return “NO”
25: return  $(\phi'_2, \phi'_3)$ 

```

Let (G, λ) be a *yes*-instance. Then, by iteratively applying Theorem 9 it follows that $\phi_{3\text{NAE}}^{(j)} \wedge \phi_{2\text{SAT}}^{(j)}$ is satisfiable, for every iteration j of the algorithm. Recall that, at the end of the last iteration k of the algorithm, $\phi_{3\text{NAE}}^{(k)} \wedge \phi_{2\text{SAT}}^{(k)}$ is empty. Then the algorithm gives the arbitrary truth value $x_i = 1$ to every variable x_i which did not yet get any truth value yet. This is a correct decision as all these variables are not involved in any Boolean constraint of $\phi_{3\text{NAE}}^{(k)} \wedge \phi_{2\text{SAT}}^{(k)}$ (which is empty). Finally, the algorithm orients all edges of G according to the corresponding truth assignment. The returned orientation F of (G, λ) is temporally transitive as every variable was assigned a truth value according to the Boolean constraints throughout the execution of the algorithm.

Now let (G, λ) be a *no*-instance. We will prove that, at some iteration $j \leq 0$, the algorithm will “NO”. Suppose otherwise that the algorithm instead returns an orientation F of (G, λ) after performing k iterations. Then clearly $\phi_{3\text{NAE}}^{(k)} \wedge \phi_{2\text{SAT}}^{(k)}$ is empty, and thus

■ **Algorithm 4** Temporal transitive orientation.

Input: A temporal graph (G, λ) , where $G = (V, E)$.

Output: A temporal transitive orientation F of (G, λ) , or the announcement that (G, λ) is temporally not transitively orientable.

```

1: Execute Algorithm 1 to build the  $\Lambda$ -implication classes  $\{A_1, A_2, \dots, A_s\}$  and the Boolean
   variables  $\{x_{uv}, x_{vu} : \{u, v\} \in E\}$ 
2: if Algorithm 1 returns “NO” then return “NO”
3: Build the 3NAE formula  $\phi_{3\text{NAE}}$  and the 2SAT formula  $\phi_{2\text{SAT}}$ 
4: if INITIAL-FORCING  $(\phi_{3\text{NAE}}, \phi_{2\text{SAT}}) \neq$  “NO” then {Initialization phase}
5:    $(\phi_{3\text{NAE}}^{(0)}, \phi_{2\text{SAT}}^{(0)}) \leftarrow$  INITIAL-FORCING  $(\phi_{3\text{NAE}}, \phi_{2\text{SAT}})$ 
6: else  $\{\phi_{3\text{NAE}} \wedge \phi_{2\text{SAT}}$  leads to a contradiction $\}$ 
7:   return “NO”
8:  $j \leftarrow 1$ ;  $F \leftarrow \emptyset$  {Main phase}
9: while a variable  $x_i$  appearing in  $\phi_{3\text{NAE}}^{(j-1)} \wedge \phi_{2\text{SAT}}^{(j-1)}$  did not yet receive a truth value do
10:  if BOOLEAN-FORCING  $(\phi_{3\text{NAE}}^{(j-1)}, \phi_{2\text{SAT}}^{(j-1)}, x_i, 1) \neq$  “NO” then
11:     $(\phi_{3\text{NAE}}^{(j)}, \phi_{2\text{SAT}}^{(j)}) \leftarrow$  BOOLEAN-FORCING  $(\phi_{3\text{NAE}}^{(j-1)}, \phi_{2\text{SAT}}^{(j-1)}, x_i, 1)$ 
12:  else  $\{x_i = 1$  leads to a contradiction $\}$ 
13:    if BOOLEAN-FORCING  $(\phi_{3\text{NAE}}^{(j-1)}, \phi_{2\text{SAT}}^{(j-1)}, x_i, 0) \neq$  “NO” then
14:       $(\phi_{3\text{NAE}}^{(j)}, \phi_{2\text{SAT}}^{(j)}) \leftarrow$  BOOLEAN-FORCING  $(\phi_{3\text{NAE}}^{(j-1)}, \phi_{2\text{SAT}}^{(j-1)}, x_i, 0)$ 
15:    else
16:      return “NO”
17:     $j \leftarrow j + 1$ 
18: for  $i = 1$  to  $s$  do
19:  if  $x_i$  did not yet receive a truth value then  $x_i \leftarrow 1$ 
20:  if  $x_i = 1$  then  $F \leftarrow F \cup A_i$  else  $F \leftarrow F \cup \overline{A_i}$ 
21: return the temporally transitive orientation  $F$  of  $(G, \lambda)$ 

```

clearly satisfiable. Therefore, iteratively applying Theorem 9 implies that $\phi_{3\text{NAE}}^{(0)} \wedge \phi_{2\text{SAT}}^{(0)}$ is also satisfiable, and thus (G, λ) is temporally transitively orientable by Observation 8, which is a contradiction to the assumption that (G, λ) be a *no*-instance.

Lastly, we prove that our algorithm runs in polynomial time. The Λ -implication classes of (G, λ) can be clearly computed in polynomial time. Our algorithm calls a subroutine BOOLEAN-FORCING at most four times for every variable in $\phi_{3\text{NAE}}^{(0)} \wedge \phi_{2\text{SAT}}^{(0)}$. BOOLEAN-FORCING iteratively adds and removes clauses from the 2SAT part of the formula, while it can only remove clauses from the 3NAE part. Whenever a clause is added to the 2SAT part, a clause of the 3NAE part is removed. Therefore, as the initial 3NAE formula has at most polynomially-many clauses, we can add clauses to the 2SAT part only polynomially-many times. Hence, we have an overall polynomial running time. ◀

4 Temporal Transitive Completion

We now study the computational complexity of TEMPORAL TRANSITIVE COMPLETION (TTC). In the static case, the so-called *minimum comparability completion* problem, i.e. adding the smallest number of edges to a static graph to turn it into a comparability graph, is known to be NP-hard [24]. Note that minimum comparability completion on static graphs is a special case of TTC and thus it follows that TTC is NP-hard too. Our other variants, however, do not generalize static comparability completion in such a straightforward way. Note that for STRICT TTC we have that the corresponding recognition problem STRICT TTO is NP-complete (Theorem 3), hence it follows directly that STRICT TTC is NP-hard. For the remaining two variants of our problem, we show in the following that they are also NP-hard, giving the result that all four variants of TTC are NP-hard. Furthermore, we present a polynomial-time algorithm for all four problem variants for the case that all edges of underlying graph are oriented, see Theorem 12. This allows directly to derive an FPT algorithm for the number of unoriented edges as a parameter.

► **Theorem 11.** *All four variants of TTC are NP-hard, even when the input temporal graph is completely unoriented.*

We now show that TTC can be solved in polynomial time, if all edges are already oriented, as the next theorem states.

► **Theorem 12.** *An instance (\mathcal{G}, F, k) of TTC where $\mathcal{G} = (G, \lambda)$ and $G = (V, E)$, can be solved in $O(m^2)$ time if F is an orientation of E , where $m = |E|$.*

Using Theorem 12 we can now prove that TTC is fixed-parameter tractable (FPT) with respect to the number of unoriented edges in the input temporal graph \mathcal{G} .

► **Corollary 13.** *Let $I = (\mathcal{G} = (G, \lambda), F, k)$ be an instance of TTC, where $G = (V, E)$. Then I can be solved in $O(2^q \cdot m^2)$, where $q = |E| - |F|$ and m the number of time edges.*

5 Deciding Multilayer Transitive Orientation

In this section we prove that MULTILAYER TRANSITIVE ORIENTATION (MTO) is NP-complete, even if every edge of the given temporal graph has at most two labels. Recall that this problem asks for an orientation F of a temporal graph $\mathcal{G} = (G, \lambda)$ (i.e. with exactly one orientation for each edge of G) such that, for every “time-layer” $t \geq 1$, the (static) oriented graph defined by the edges having time-label t is transitively oriented in F . As we discussed in Section 2, this problem makes more sense when every edge of G potentially has multiple time-labels, therefore we assume here that the time-labeling function is $\lambda : E \rightarrow 2^{\mathbb{N}}$.

► **Theorem 14.** *MTO is NP-complete, even on temporal graphs with at most two labels per edge.*

References

- 1 Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *Journal of Parallel and Distributed Computing*, 87:109–120, 2016.
- 2 Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017.

- 3 Eleni C. Akrida, George B. Mertzios, Sotiris E. Nikolettseas, Christoforos L. Raptopoulos, Paul G. Spirakis, and Viktor Zamaraev. How fast can we reach a target vertex in stochastic temporal graphs? *Journal of Computer and System Sciences*, 114:65–83, 2020. An extended abstract appeared at ICALP 2019.
- 4 Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Viktor Zamaraev. Temporal vertex cover with a sliding time window. *Journal of Computer and System Sciences*, 107:108–123, 2020.
- 5 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539, 2021.
- 6 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.
- 7 Kyriakos Axiotis and Dimitris Fotakis. On the size and the approximability of minimum temporally connected subgraphs. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 149:1–149:14, 2016.
- 8 Matthias Bentert, Anne-Sophie Himmel, Hendrik Molter, Marco Morik, Rolf Niedermeier, and René Saitenmacher. Listing all maximal k -plexes in temporal graphs. *ACM Journal of Experimental Algorithmics*, 24(1):13:1–13:27, 2019.
- 9 Matthias Bentert, Anne-Sophie Himmel, André Nichterlein, and Rolf Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. *Applied Network Science*, 5(1):73, 2020.
- 10 Robert Bredereck, Christian Komusiewicz, Stefan Kratsch, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. Assessing the computational complexity of multilayer subgraph detection. *Network Science*, 7(2):215–241, 2019.
- 11 Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.
- 12 Sebastian Buß, Hendrik Molter, Rolf Niedermeier, and Maciej Rymar. Algorithmic aspects of temporal betweenness. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2084–2092. ACM, 2020.
- 13 Arnaud Casteigts and Paola Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, April 2013. URL: <https://hal.archives-ouvertes.fr/hal-00865762>.
- 14 Arnaud Casteigts and Paola Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada, April 2013. URL: <https://hal.archives-ouvertes.fr/hal-00865764>.
- 15 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.
- 16 Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. In *31st International Symposium on Algorithms and Computation (ISAAC)*, pages 30:1–30:18, 2020.
- 17 Arnaud Casteigts, Joseph G. Peters, and Jason Schoeters. Temporal cliques admit sparse spanners. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 134:1–134:14, 2019.
- 18 Jiehua Chen, Hendrik Molter, Manuel Sorge, and Ondřej Suchý. Cluster editing in multi-layer and temporal graphs. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2018.
- 19 J. Enright, K. Meeks, G.B. Mertzios, and V. Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021.

- 20 Jessica Enright, Kitty Meeks, and Fiona Skerman. Assigning times to minimise reachability in temporal graphs. *Journal of Computer and System Sciences*, 115:169–186, 2021.
- 21 Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 444–455, 2015.
- 22 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020.
- 23 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2nd edition, 2004.
- 24 S Louis Hakimi, Edward F Schmeichel, and Neal E Young. Orienting graphs to optimize reachability. *Information Processing Letters*, 63(5):229–235, 1997.
- 25 Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. Adapting the Bron-Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35:1–35:16, 2017.
- 26 Petter Holme and Jari Saramäki. *Temporal network theory*, volume 2. Springer, 2019.
- 27 David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.
- 28 Hyoungshick Kim and Ross Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85(2):026107, 2012.
- 29 Ross M. McConnell and Jeremy P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 536–545, 1994.
- 30 Ross M. McConnell and Jeremy P. Spinrad. Linear-time transitive orientation. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 19–25, 1997.
- 31 Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3):189–241, 1999.
- 32 David B McDonald and Daizaburo Shizuka. Comparative transitive and temporal orderliness in dominance networks. *Behavioral Ecology*, 24(2):511–520, 2013.
- 33 George B. Mertzios. The recognition of simple-triangle graphs and of linear-interval orders is polynomial. *SIAM Journal on Discrete Mathematics*, 29(3):1150–1185, 2015.
- 34 George B. Mertzios, Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 657–668, 2013.
- 35 George B Mertzios, Hendrik Molter, Rolf Niedermeier, Viktor Zamaraev, and Philipp Zschoche. Computing maximum matchings in temporal graphs. In *Proceedings of the 37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 154, pages 27:1–27:14, 2020.
- 36 George B. Mertzios, Hendrik Molter, Malte Renken, Paul G. Spirakis, and Philipp Zschoche. The complexity of transitively orienting temporal graphs. *arXiv preprint*, 2021. [arXiv: 2102.06783](https://arxiv.org/abs/2102.06783).
- 37 George B Mertzios, Hendrik Molter, and Viktor Zamaraev. Sliding window temporal graph coloring. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 7667–7674, 2019.
- 38 Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72–72, 2018.
- 39 Robert Moskovitch and Yuval Shahar. Medical temporal-knowledge discovery via temporal abstraction. In *Proceedings of the AMIA Annual Symposium*, page 452, 2009.
- 40 Robert Moskovitch and Yuval Shahar. Fast time intervals mining using the transitivity of temporal relations. *Knowledge and Information Systems*, 42(1):21–48, 2015.

75:18 The Complexity of Transitively Orienting Temporal Graphs

- 41 V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo, and V. Latora. Graph metrics for temporal networks. In *Temporal Networks*. Springer, 2013.
- 42 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.
- 43 Jeremy P. Spinrad. On comparability and permutation graphs. *SIAM Journal on Computing*, 14(3):658–670, 1985.
- 44 Jeremy P. Spinrad. *Efficient graph representations*, volume 19 of *Fields Institute Monographs*. American Mathematical Society, 2003.
- 45 Xavier Tannier and Philippe Muller. Evaluating temporal graphs built from texts via transitive reduction. *Journal of Artificial Intelligence Research (JAIR)*, 40:375–413, 2011.
- 46 Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.
- 47 Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016.
- 48 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020.

Temporal Reachability Minimization: Delaying vs. Deleting

Hendrik Molter  

Department of Industrial Engineering and Management, Ben-Gurion University of the Negev,
Beer Sheva, Israel

Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Malte Renken  

Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Philipp Zschoche  

Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Germany

Abstract

We study spreading processes in temporal graphs, i. e., graphs whose connections change over time. These processes naturally model real-world phenomena such as infectious diseases or information flows. More precisely, we investigate how such a spreading process, emerging from a given set of sources, can be contained to a small part of the graph. To this end we consider two ways of modifying the graph, which are (1) deleting connections and (2) delaying connections. We show a close relationship between the two associated problems and give a polynomial time algorithm when the graph has tree structure. For the general version, we consider parameterization by the number of vertices to which the spread is contained. Surprisingly, we prove W[1]-hardness for the deletion variant but fixed-parameter tractability for the delaying variant.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Paths and connectivity problems; Mathematics of computing → Network flows

Keywords and phrases Temporal Graphs, Temporal Paths, Disease Spreading, Network Flows, Parameterized Algorithms, NP-hard Problems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.76

Funding *Hendrik Molter*: Supported by the German Research Foundation (DFG), project MATE (NI 369/17), and by the Israeli Science Foundation (ISF), grant No. 1070/20. The main work was done while affiliated with TU Berlin.

Malte Renken: Supported by the German Research Foundation (DFG), project MATE (NI 369/17).

Acknowledgements This work was initiated at the research retreat of the Algorithmics and Computational Complexity group of TU Berlin in September 2020 in Zinnowitz.

1 Introduction

Reachability is a fundamental problem in graph theory and algorithmics [38, 39, 33, 17] and quite well-understood. With the emergence of temporal graphs,¹ the concept of reachability was extended to the dimension of time using *temporal paths* [34, 6]. For a vertex s to reach another vertex z in a temporal graph, there must not only be a path between them but the edges of this path have to appear in chronological order. This requirement makes temporal reachability non-symmetric and non-transitive, which stands in contrast to reachability in normal (static) graphs. Reachability is arguably one of the most central concepts in temporal graph algorithmics and has been studied under various aspects, such as path finding [40, 5, 9, 11], vertex separation [34, 28, 41], finding spanning subgraphs [12, 4], temporal graph exploration [25, 23, 26, 7, 24, 3], and others [35, 2, 10, 32].

¹ Temporal graphs are graphs whose edge set changes over discrete time.



Perhaps the most prominent application of temporal graph reachability is currently epidemiology, dealing with effective prevention or containment of disease spreading [1]. Here, minimizing the reachability of vertices in a temporal graph by manipulating the temporal graph corresponds to minimizing the spread of an infection in various networks by some countermeasures. Application instances for this scenario may be drawn from physical contacts [27, 19] or airline flights [8, 13], but also social networks [31, 14], cattle movements [36], or computer networks [37].

Enright et al. [21] studied the problem of deleting k time-edges² such that no single vertex can reach more than r other vertices and showed its NP-hardness and W[1]-hardness for the parameter k , even in very restricted settings. Here, we shift the focus to a set of multiple given sources, thus studying the following problem, which has not been considered for computational complexity analysis yet (to the best of our knowledge).

MINIMIZING TEMPORAL REACHABILITY BY DELETING (MINREACHDELETE)

Input: A temporal graph \mathcal{G} , a set of sources vertices S , and integers k, r .

Question: Can we delete at most k time-edges s.t. at most r vertices are reachable from S ?

Imaginably, removing edges or vertices is not the most *infrastructure friendly* approach to restrict reachability. To address this, other operations have been studied. Enright et al. [22] considered restricting the reachability by just changing the relative order in which edges are active. Deligkas and Potapov [15] considered restricting the reachability by a merging operation of consecutive edge sets of the temporal graph and by a delay operation of time-edges by δ time steps, i.e., moving a time-edge from time t to $t + \delta$. In particular, they introduced a delay variant of MINREACHDELETE.

MINIMIZING TEMPORAL REACHABILITY BY DELAYING (MINREACHDELAY)

Input: A temporal graph \mathcal{G} , a set of sources vertices $S \subseteq V$, and integers k, r, δ .

Question: Can we delay at most k time-edges by δ s.t. at most r vertices are reachable from S ?

This is the central problem studied in this paper. Throughout the whole paper we assume for all instances of MINREACHDELETE and MINREACHDELAY that $0 < |S| \leq r$. We remark that technically Deligkas and Potapov [15] formulate the problem slightly differently, allowing delays of *up to* δ to appear. However, a simple argument can be given to see that this distinction is not significant: Clearly, delaying a time-edge reduces the number of reachable vertices only if the undelayed time-edge could be reached from some source $s \in S$. But when this is the case, increasing the delay of that time-edge can never increase the set of vertices reachable from S . The reason is that, while increasing the delay might enable some source $s' \in S \setminus \{s\}$ to reach a vertex v , that vertex v would be reached from s in any case.

Deligkas and Potapov [15] showed that MINREACHDELAY is NP-hard and W[1]-hard when parameterized by k , even if underlying graph has lifetime $\tau = 2$. A close look into the proof reveals that this also holds for MINREACHDELETE.

Our contribution. We study how MINREACHDELETE and MINREACHDELAY relate to each other. We show that both problems are polynomial-time solvable on trees. Moreover, there is an intermediate reduction from MINREACHDELETE to MINREACHDELAY indicating that MINREACHDELAY seems generally harder than MINREACHDELETE. However, surprisingly, this is no longer true when we parameterize the problems by the number r of reachable vertices. Here, we develop a max-flow-based branching strategy and obtain fixed-parameter

² That is, an edge at a point in time.

tractability for MINREACHDELAY while MINREACHDELETE remains W[1]-hard. This makes MINREACHDELAY particularly interesting for applications where the number of reachable vertices should be very small, e.g. when trying to contain the spread of dangerous diseases.

2 Preliminaries

We define \mathbb{N} as the positive natural numbers, $[a, b] := \{i \in \mathbb{Z} \mid a \leq i \leq b\}$, and $[n] := [1, n]$. For a function $f: V \rightarrow \mathbb{Z}$ and subset $X \subseteq V$ we denote by $f(X)$ the sum $\sum_{x \in V} f(x)$. We use standard notation from graph theory [16]. We say for a (directed) graph G that $G - X := G[V(G) \setminus X]$ is the induced subgraph of G when the vertices in X are removed, and $G \setminus Y := (V(G), E(G) \setminus Y)$ is the subgraph when the edges in Y are removed, where X is a vertex set and Y is an edge set. For any predicate P , the Iverson bracket $[P]$ is 1 if P is true and 0 otherwise.

Parameterized complexity. Let Σ denote a finite alphabet. A parameterized problem $L \subseteq \{(x, k) \in \Sigma^* \times \mathbb{N} \cup \{0\}\}$ is a subset of all instances (x, k) from $\Sigma^* \times \mathbb{N} \cup \{0\}$, where k denotes the *parameter*. A parameterized problem L is in FPT (is *fixed-parameter tractable*) if there is an algorithm that decides every instance (x, k) for L in $f(k) \cdot |x|^{O(1)}$ time, where f is any computable function only depending on the parameter. If a parameterized problem L is W[1]-hard, then it is presumably not fixed-parameter tractable. We refer to Downey and Fellows [18] for details.

Temporal graphs. A *temporal graph* \mathcal{G} consists of a set of vertices V (or $V(\mathcal{G})$), and a sequence of edge sets $(E_i)_{i \in [\tau]}$ where each E_i is a set of unordered pairs from V . The number τ is called the *lifetime* of \mathcal{G} . The elements of $\mathcal{E}(\mathcal{G}) := \bigcup_{i \in [\tau]} E_i \times \{i\}$ are called the *time-edges* of \mathcal{G} . Furthermore \mathcal{G} has a *traversal time* function $\gamma: \mathcal{E}(\mathcal{G}) \rightarrow \mathbb{N}$ specifying the time it takes to traverse each time-edge. The temporal graph \mathcal{G} is then written as the tuple $(V, (E_i)_{i \in [\tau]}, \gamma)$. Often we assume γ to be the constant function $\gamma \equiv 1$ and then simply write $\mathcal{G} = (V, (E_i)_{i \in [\tau]})$. The *underlying graph* of \mathcal{G} is the graph $(V, \bigcup_{i=1}^{\tau} E_i)$. For a time-edge set Y and temporal graph \mathcal{G} , we denote by $\mathcal{G} \setminus Y$ the temporal graph where $V(\mathcal{G} \setminus Y) = V(\mathcal{G})$ and $\mathcal{E}(\mathcal{G} \setminus Y) = \mathcal{E}(\mathcal{G}) \setminus Y$. A *temporal s-z-path* in \mathcal{G} is a sequence of time-edges $P = (e_i = (\{v_{i-1}, v_i\}, t_i))_{i=1}^m$ where

1. $v_0 = s$ and $v_m = z$,
2. the sequence of edges $(\{v_{i-1}, v_i\})_{i=1}^m$ forms an *s-z-path* in the underlying graph of \mathcal{G} , and
3. $t_{i+1} \geq t_i + \gamma(e_i)$ for all $i \in [m - 1]$.

The *arrival time* of P is $t_m + \gamma(e_m)$. The set of vertices of P is denoted by $V(P) = \{v_i \mid 0 \leq i \leq m\}$. A vertex w is *reachable* from v in \mathcal{G} (at time t) if there exists a temporal v - w -path in \mathcal{G} (with arrival time at most t). In particular, every vertex reaches itself via a trivial path. Furthermore, w is reachable from $S \subseteq V$ if there is a temporal s - w -path for some $s \in S$, and the set of all vertices reachable from S is denoted the *reachable set* $R_{\mathcal{G}}(S)$. We drop the index \mathcal{G} if it is clear from the context. *Delaying* a time-edge $(\{v, w\}, t)$ by δ refers to replacing it with the time-edge $(\{v, w\}, t + \delta)$. For a temporal graph \mathcal{G} and a time-edge set $X \subseteq \mathcal{E}(\mathcal{G})$ we denote by $\mathcal{G} \nearrow_{\delta} X$ the temporal graph \mathcal{G} where the time-edges in X are delayed by δ .

Preliminary observations. We present an intermediate polynomial-time reduction from the MINREACHDELETE to MINREACHDELAY.

► **Lemma 1.** *Given an instance $I = (\mathcal{G} = (V, (E_i)_{i \in [\tau]}, S, k, r)$ of MINREACHDELETE , we can compute in linear time, an instance $J = (\mathcal{G}' = (V', (E'_i)_{i=1}^{6\tau+1}), S', k, r', \delta = 3\tau)$ of MINREACHDELAY such that the feedback vertex number³ of the underlying graph of \mathcal{G} and \mathcal{G}' is the same, and I is a yes-instance if and only if J is a yes-instance.*

Proof. We construct $\mathcal{G}' = (V', (E'_i)_{i=1}^{3\tau+\delta+1})$ in the following way. Set

$$\begin{aligned} V_e &:= \{e_{uv}, e_{vu} \mid (\{v, u\}, t) \in \mathcal{E}(\mathcal{G})\}, & V_s &:= \{s_{vu} \mid e_{vu} \in V_e\}, \\ V' &:= V \cup V_e \cup V_s, \text{ and} & S' &:= S \cup V_s. \end{aligned}$$

Begin with $E'_i = \emptyset$ for all $i \in [3\tau]$. Then, add for each time-edge $(\{v, u\}, t) \in \mathcal{E}(\mathcal{G})$, the time-edges $(\{v, e_{vu}\}, 3t - 2)$, $(\{v, e_{vu}\}, 3t)$, $(\{e_{vu}, e_{uv}\}, 3t - 1)$, $(\{u, e_{uv}\}, 3t - 2)$, and $(\{u, e_{uv}\}, 3t)$ to \mathcal{G}' . Afterwards, add the time-edge $(\{s_{vu}, e_{vu}\}, 6\tau + 1 = 3\tau + \delta + 1)$ for each $e_{vu} \in V_e$. Finally we set $r' := r + |V' \setminus V|$. Since we add for each time-edge of \mathcal{G} a constant number of vertices and time-edges to \mathcal{G}' , we have that $|\mathcal{G}'| \in O(|\mathcal{G}|)$ and \mathcal{G}' can be computed in linear time. Moreover, the underlying graph G' of \mathcal{G}' is obtained from the underlying graph G of \mathcal{G} by subdividing edges and adding leaves, thus G and G' have the same feedback vertex number. It remains to prove that the two instances are equivalent.

(\Rightarrow): Let X be a solution for I . Then, set $X' := \{(\{e_{vu}, e_{uv}\}, 3t - 1) \mid (\{v, u\}, t) \in X\}$. Pick $s \in S$ and $v \in V$ arbitrary. Note that for each temporal s - v -path P' in \mathcal{G}' , we can construct a temporal s - v -path P in \mathcal{G} which uses a time-edge $(\{u, w\}, t)$ if and only if P' uses the time-edge $(\{e_{uw}, e_{wu}\}, 3t - 1)$. Furthermore, any temporal s - v -path in $\mathcal{G}' \nearrow_\delta X'$ cannot use any delayed time-edge. As s and v are arbitrary, this proves $R_{\mathcal{G}' \nearrow_\delta X'}(S') \cap V \subseteq R_{\mathcal{G} \setminus X}(S)$. Thus, at most $r + |V' \setminus V| = r'$ vertices are reachable from S' in $\mathcal{G}' \nearrow_\delta X'$, thus J is a yes-instance.

(\Leftarrow): Let X' be a solution for J . Begin by observing that delaying any time-edges between V_s and V_e has no effect. Consequently, $R_{\mathcal{G}' \nearrow_\delta X'}(V_s) = V_s \cup V_e$ for every possible choice of X' . Therefore X' is a valid solution if and only if $|R_{\mathcal{G}' \nearrow_\delta X'}(S) \cap V| \leq r' - |V_s \cup V_e| = r$, i.e., we only need to study the reachability between vertices in V . Because of this, delaying a time-edge connecting two vertices of V_e has the same effect as deleting that time-edge. Next, observe that instead of delaying some time-edge $(\{v, e_{vu}\}, t)$ which connects vertices of V and V_e , the same or better reduction of reachability is achieved by instead delaying $(\{e_{vu}, e_{uv}\}, t')$, with $t' \in \{t - 1, t + 1\}$ chosen appropriately. Due to this, we may assume without loss of generality that $X' \subseteq \{(\{e_{vu}, e_{uv}\}, 3t - 1) \mid (\{v, u\}, t) \in \mathcal{E}(\mathcal{G})\}$.

Set $X := \{(\{v, u\}, t) \mid (\{e_{uv}, e_{vu}\}, 3t - 1) \in X'\}$. Let $s \in S$ and $v \in V$ be arbitrary. Note that for each temporal s - v -path P in \mathcal{G} , we can construct a temporal s - v -path P' in \mathcal{G}' as above. Thus $R_{\mathcal{G} \setminus X}(S) \subseteq R_{\mathcal{G}' \nearrow_\delta X'}(S) \cap V$. Since we already observed that $|R_{\mathcal{G}' \nearrow_\delta X'}(S') \cap V| \leq r$, we conclude that I is a yes-instance. ◀

Note that the reduction in Lemma 1 preserves the size k of the solution and the feedback vertex number of the underlying graph. We remark that, in exchange for dropping the latter property, one can modify the reduction to instead have $|S'| = |S|$, by simply adding time-edges from S to V_s before all other time-edges. However, in any case the size r' of the reachable set in J is unbounded in terms of the size r of the reachable set in I . Unless $\text{FPT} = \text{W}[1]$, this is unavoidable as we learn in the next section.

³ That is, the minimum number of vertices needed to hit all cycles in an undirected graph.

3 Parameterized by the Reachable Set Size

In this section the study MINREACHDELAY and MINREACHDELETE parameterized by the reachable set size r . In particular, our main result in this section is the fixed-parameter tractability of MINREACHDELAY parameterized by r . This is in stark contrast to the $W[1]$ -hardness of MINREACHDELETE parameterized by r which we show first.

► **Theorem 2.** *MINREACHDELETE parameterized by r is $W[1]$ -hard, even if $\tau = 2$.*

Proof. We present a parameterized reduction from the $W[1]$ -hard [18] CLIQUE problem parameterized by ℓ , where given a graph $H = (U, F)$ we are asked whether H contains a clique of size ℓ .

Let $H = (U, F)$ be a graph, where $|F| = m$. We construct an instance $I = (\mathcal{G}, \{s\}, k = m - \binom{\ell}{2}, r = 1 + \ell + \binom{\ell}{2})$ of MINREACHDELETE , where $\mathcal{G} := (V, (E_i)_{i \in [2]})$ is the temporal graph given by

$$\begin{aligned} V &:= U \cup \{s\} \cup \{e_f \mid f \in F\}, \\ E_1 &:= \{\{s, e_f\} \mid f \in F\}, \quad \text{and} \quad E_2 := \{\{e_{\{u,v\}}, u\}, \{e_{\{u,v\}}, v\} \mid \{u, v\} \in F\}. \end{aligned}$$

Note that I can be constructed in polynomial time.

(\Rightarrow): Let $C = (V', F')$ be a clique of size ℓ in H . We set $X := \{(\{s, e_f\}, 1) \mid f \in F \setminus F'\}$. Note that $|X| \leq k$ and that for each edge $f \in F$ we can reach e_f from s if and only if $f \in F'$. Hence, by the construction of \mathcal{G} , a vertex $u \in U$ is reachable from s in $\mathcal{G} \setminus X$ if and only if $u \in V'$. Hence, we can reach $1 + \binom{\ell}{2} + \ell$ many vertices from s in $\mathcal{G} \setminus X$. Thus, I is a *yes*-instance.

(\Leftarrow): Let $X \subseteq \mathcal{E}(\mathcal{G})$ be a solution for I . Without loss of generality, we can assume that X does not contain a time-edge $(\{e_f, u\}, 2)$, because it can be replaced by $(\{e_f, s\}, 1)$. Observe that at least $\binom{\ell}{2}$ vertices from $\{e_f \mid f \in F\}$ are reachable from s in $\mathcal{G} \setminus X$. Since $r = 1 + \binom{\ell}{2} + \ell$, we can reach from s at most ℓ vertices from U . Hence, $U \cap R_{\mathcal{G} \setminus X}(\{s\})$ must form a clique of size ℓ in H . ◀

Due to Theorem 2, we know that there is presumably no $f(r) \cdot |\mathcal{G}|^{O(1)}$ -time algorithm to decide whether we can keep the reachable set of a vertex s of \mathcal{G} small (at most r vertices), by deleting at most k time-edges. However, this changes when we delay (instead of deleting) at most k edges. Formally, we show the following.

► **Theorem 3.** *MINREACHDELAY is solvable in $O(r! \cdot k \cdot |\mathcal{G}|)$ time.*

The proof of Theorem 3 is structured as follows.

Step 1 (reduction to slowing): We reduce MINREACHDELAY to an auxiliary problem which we call MINREACHSLOW . Here, instead of *delaying* a time-edge (moving it δ layers forward in time) we *slow* it, i. e., increase the time required to traverse it by δ .

Step 2 (flow-based techniques): Our new target now is a fixed-parameter algorithm for MINREACHSLOW . Since we do not aim to preserve a specific temporal graph class, we simplify the input by replacing S with a single-source s . Then we transform the temporal graph \mathcal{G} into a (non-temporal) directed graph D in which the deletion of an edge corresponds to slowing a temporal edge in \mathcal{G} . Using this, we derive a max-flow-based polynomial-time algorithm which checks whether the source s can be prevented from reaching any vertices outside of a given set R by slowing at most k time-edges in \mathcal{G} .

Step 3 (resulting search-tree): We are aiming for a search-tree algorithm for MINREACHSLOW. Let R be a set of vertices and suppose that our max-flow-based algorithm failed to prevent s from reaching any vertices outside of R . Now, if there exists a solution for the given instance of MINREACHSLOW, then we can identify less than $|R|$ vertices such that at least one of them will be always reached from s . We can then try adding each of them to R , gradually building a search-tree to find the solution.

Henceforth the details follow. Instead of solving MINREACHDELAY directly, we reduce it to an auxiliary problem introduced next. Let $\mathcal{G} = (V, (E_i)_{i \in [\tau]}, \gamma)$ be a temporal graph. *Slowing* a time-edge $(\{v, w\}, t)$ by δ refers to increasing $\gamma(\{v, w\}, t)$ by δ . We define $\mathcal{G} \uparrow_\delta X := (V, (E_i)_{i \in [\tau]}, \gamma')$ where $\gamma'(e) := \gamma(e) + \delta \cdot [e \in X]$. Our auxiliary problem is the following.

MINIMIZING TEMPORAL REACHABILITY BY SLOWING (MINREACHSLOW)

Input: A temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\tau]}, \gamma)$, a set of sources $S \subseteq V$, and integers k, r, δ .

Question: Is there a time-edge set $X \subseteq \mathcal{E}(\mathcal{G})$ of size at most k such that $|R_{\mathcal{G} \uparrow_\delta X}(S)| \leq r$?

By the following, solving an instance of MINREACHSLOW also solves MINREACHDELAY.

► **Lemma 4.** *An instance $I = (\mathcal{G} = (V, (E_i)_{i \in [\tau]}, \gamma), S, k, r, \delta)$ of MINREACHDELAY is a yes-instance if and only if $J = (\mathcal{G}, S, k, r, \delta)$ is a yes-instance of MINREACHSLOW.*

Proof.

(\Rightarrow): Let $X \subseteq \mathcal{E}(\mathcal{G})$ be a solution for I . Note that for every temporal v - w -path P in $\mathcal{G} \uparrow_\delta X$, we can construct a temporal v - w -path P' in $\mathcal{G} \nearrow_\delta X$ by replacing a time-edge $(e, t) \in P$ with $(e, t + \delta)$ whenever $(e, t) \in X$. Hence, the reachable set of $s \in S$ in $\mathcal{G} \nearrow_\delta X$ is a superset of the reachable set of s in $\mathcal{G} \uparrow_\delta X$. Thus, J is a yes-instance.

(\Leftarrow): Let $X \subseteq \mathcal{E}(\mathcal{G})$ be an inclusion-minimal solution for J . We claim that every vertex reachable from S in $\mathcal{G} \nearrow_\delta X$ by some time t is also reachable from S in $\mathcal{G} \uparrow_\delta X$ until time t . Suppose for contradiction that the claim does not hold true for some vertex z and let t be the time S reaches z in $\mathcal{G} \nearrow_\delta X$. We may assume z to be chosen to minimize t . Clearly $t > 0$, i. e., $z \notin S$. Let P be a temporal s - z -path in $\mathcal{G} \nearrow_\delta X$ with arrival time t and $s \in S$. Let u be the penultimate vertex of P and $(\{u, z\}, t')$ the last time-edge of P . By minimality of t , u must be reachable from S by time t' also in $\mathcal{G} \uparrow_\delta X$. Since all time-edges of $\mathcal{E}(\mathcal{G}) \setminus X$ appear in $\mathcal{G} \nearrow_\delta X$ and $\mathcal{G} \uparrow_\delta X$ with identical traversal times, the last time-edge $(\{u, z\}, t')$ of P must be in $\mathcal{E}(\mathcal{G} \nearrow_\delta X) \setminus (\mathcal{E}(\mathcal{G}) \setminus X)$. Thus $(\{u, z\}, t' - \delta) \in X$. By minimality of X , there must be a source $s' \in S$ and a temporal s' - u -path P' in $\mathcal{G} \uparrow_\delta X$ reaching either u or z at time $t' - \delta$. If P' reaches z , then this is clearly a contradiction. But if P' reaches u , then appending $(\{u, z\}, t - \delta)$ to P' produces a temporal s' - z -path in $\mathcal{G} \uparrow_\delta X$ arriving at time t , thus also a contradiction. ◀

In the remainder of this section, we show that MINREACHSLOW is fixed-parameter tractable, when parameterized by r . Formally, we aim for the following theorem, which in turn clearly implies Theorem 3 by the means of Lemma 4.

► **Theorem 5.** *MINREACHSLOW can be solved in $O(r! \cdot k \cdot |\mathcal{G}|)$ time.*

The remainder of this section is dedicated to proving Theorem 5. The advantage of considering MINREACHSLOW instead of MINREACHDELAY is that we do not have to deal with new time-edges appearing due to the delay operation. This allows us to translate the reachability of a temporal graph to a (non-temporal) directed graph specially tailored to MINREACHSLOW.

In particular, the removal of some edges in the directed graph corresponds to slowing the corresponding time-edges by δ in the temporal graph. Before giving the details of the construction, we first reduce to the case where S is a singleton.

► **Lemma 6.** *Given an instance $I = (\mathcal{G} = (V, (E_i)_{i \in [\tau]}, S, k, r, \delta)$ of MINREACHSLOW , we can construct in linear time a instance $J = (\mathcal{G}', \{s\}, k, r + 1, \delta)$ of MINREACHSLOW such that I is a yes-instance if and only if J is a yes-instance.*

Proof. We set $\mathcal{G}' := (V \cup \{s\}, (E'_i)_{i \in [\tau + \delta + 1]})$ where s is a new vertex, $E'_1 := \{\{s, s'\} \mid s' \in S\}$, $E'_i := \emptyset$ for all $i \in [\delta + 1] \setminus \{1\}$, and $E'_{i + \delta + 1} := E_i$ for all $i \in [\tau]$. Observe that slowing an edge in E'_1 has no effect. Thus, I is a yes-instance if and only if J is a yes-instance. Clearly, J can be computed in linear time. ◀

A network (D, c) consists of a directed graph $D = (V, A)$ and edge capacities $c: A \rightarrow \mathbb{N}_0 \cup \{\infty\}$. A function $f: A \rightarrow \mathbb{N} \cup \{0\}$ is an s - z -flow for two distinct vertices $s, z \in V$ if

- $\forall e \in A: f(e) \leq c(e)$ and
- $\forall v \in V \setminus \{s, z\}: \sum_{(u,v) \in A} f((u,v)) = \sum_{(v,u) \in A} f((v,u))$.

The value of f is denoted by $|f| := \sum_{(s,v) \in A} f((s,v))$. An arc set $C \subseteq A$ is an s - z -cut of a network $((V, A), c)$ if $s, z \in V$ and there is no s - z -path in $(V, A \setminus C)$. The capacity of the s - z -cut C is $c(C) := \sum_{e \in C} c(e)$.

Let $\mathcal{G} = (V, (E_i)_{i \in [\tau]}, \gamma)$ be a temporal graph. We define the temporal neighborhood of a vertex $v \in V$ at time point $t \in [\tau]$ as the set $N_{\mathcal{G}}(v, t) := \bigcup_{i=t}^{\tau} N_{(V, E_i)}(v)$ containing all neighbors of v in the layers t through τ .

For any $s \in R \subseteq V$ and $\delta \in \mathbb{N}$, we define the flow network $\mathcal{F}(\mathcal{G}, s, R, \delta) := (D, c)$ where $D = (V', A)$ is the directed graph defined by

$$V' := \{s^0, z\} \cup \left\{ \begin{array}{l} e_1, e_2, v^t, w^t, v^{t+\gamma(e)}, w^{t+\gamma(e)}, \\ v^{t+\gamma(e)+\delta}, w^{t+\gamma(e)+\delta} \end{array} \mid v, w \in R \text{ and } e = (\{v, w\}, t) \in \mathcal{E}(\mathcal{G}) \right\}$$

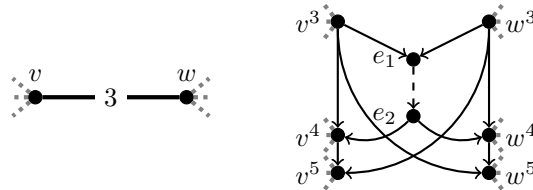
and

$$A := \left\{ (v^t, v^{t'}) \mid v^t \in V', t' = \min\{i \mid i > t \text{ and } v^i \in V'\} \neq \infty \right\} \quad (1)$$

$$\cup \left\{ \begin{array}{l} (v^t, e_1), (w^t, e_1), (e_1, e_2), \\ (e_2, v^{t+\gamma(e)}), (e_2, w^{t+\gamma(e)}), \\ (v^t, w^{t+\gamma(e)+\delta}), (w^t, v^{t+\gamma(e)+\delta}) \end{array} \mid v, w \in R \text{ and } e = (\{v, w\}, t) \in \mathcal{E}(\mathcal{G}) \right\} \quad (2)$$

$$\cup \{(v^t, z) \mid v \in R, t = \max\{i \mid N_{\mathcal{G}}(v, i) \not\subseteq R\} \neq -\infty\}. \quad (3)$$

and we set $c((e_1, e_2)) = 1$ for all $e \in \mathcal{E}(\mathcal{G})$ and $c(a) = \infty$ for all other $a \in A$. Consider Figure 1 for an illustration.



■ **Figure 1** Left: An excerpt of a temporal graph \mathcal{G} containing the time-edge $e = (\{v, w\}, 3)$ and $\gamma(e) = 1$. Right: An excerpt of the flow network $\mathcal{F}(\mathcal{G}, s, R, \delta = 1)$ showing the corresponding part for e , where solid arcs have capacity ∞ and the dashed arc has capacity 1.

► **Lemma 7.** For any given $\mathcal{G} = (V, (E_i)_{i \in [\tau]}, \gamma)$, $s \in R \subseteq V$, and $k, \delta \in \mathbb{N}$, one can test in $O(k \cdot |\mathcal{G}|)$ time whether $\mathcal{F}(\mathcal{G}, s, R, \delta)$ has a s^0 -z-flow of value at least $k + 1$ and compute such a flow or a maximum flow otherwise.

Proof. Note, that the flow network $\mathcal{F}(\mathcal{G}, s, R, \delta)$ can be computed in $O(|\mathcal{G}|)$ time by iterating over $\mathcal{E}(\mathcal{G})$ first forward and then backwards once. Then we can compute a flow of value $k + 1$ or of maximum value, whichever is smaller, by running at most $k + 1$ rounds of the Ford-Fulkerson algorithm [29]. This gives a overall running time of $O(k \cdot |\mathcal{G}|)$ time. ◀

► **Lemma 8.** Let $\mathcal{G} = (V, (E_i)_{i \in [\tau]}, \gamma)$, $s \in R \subseteq V$, $\delta \in \mathbb{N}$, and $((V', A), c) := \mathcal{F}(\mathcal{G}, s, R, \delta)$. Let $X \subseteq \mathcal{E}(\mathcal{G})$ and $C := \{(e_1, e_2) \in A \mid e \in X\}$. For any $x^t \in V'$, there is a s^0 - x^t -path in $(V', A \setminus C)$ if and only if $\mathcal{G} \uparrow_\delta X$ contains a temporal s - x -path with arrival time at most t .

Proof. Let γ' be the traversal time function of $\mathcal{G} \uparrow_\delta X$, i. e., $\gamma'(e) := \gamma(e) + \delta \cdot [e \in X]$.

(\Leftarrow): Let P be a temporal s - x -path in $\mathcal{G} \uparrow_\delta X$ for some vertex $x \in V$ and let t be the arrival time of P . Then we construct a s^0 - x^t -path \hat{P} in $(V', A \setminus C)$ as follows. Start with \hat{P} being just the vertex s^0 and perform the following two steps for every time-edge $e = (\{v, w\}, b)$ of P in order.

1. Note that the currently last vertex of \hat{P} is v^c for some $c \leq b$. As long as $c < b$, append to \hat{P} the arc (v^c, v^d) where $d > c$ is chosen minimal (cf. (1)). Afterwards, the currently last vertex of \hat{P} is v^b .
2. If $e \notin X$, i. e., $(e_1, e_2) \notin C$, then append to \hat{P} the arcs (v^b, e_1) , (e_1, e_2) , and $(e_2, w^{b+\gamma(e)})$ (cf. (2)). Otherwise, if $e \in X$, i. e., $\gamma'(e) = \gamma(e) + \delta$, then append to \hat{P} the arc $(v^b, w^{b+\gamma(e)+\delta})$. Note that in both cases the new last vertex of \hat{P} is $w^{b+\gamma'(e)}$.

(\Rightarrow): Let P be a s^0 - x^t -path in $(V', A \setminus C)$. Then we construct a s - x -path P' with arrival time at most t in $\mathcal{G} \uparrow_\delta X$ as follows. Start with P' being just the vertex s (with arrival time 0) and repeat the following steps until all arcs of P have been processed.

1. Let b be the arrival time of P' and v the last vertex. Note that the last vertex of P is v^c for some $c \geq b$. Ignore all arcs of P up to the last arc containing v^c for some $c \geq b$.
2. If the next three arcs in P are (v^c, e_1) , (e_1, e_2) , $(e_2, w^{c+\gamma(e)})$ for some time-edge $e = (\{v, w\}, c) \in \mathcal{E}(\mathcal{G})$, then append to P' that time-edge e . Note that, by assumption, $(e_1, e_2) \notin C$, thus $e \notin X$, and thus the arrival time of e is $c + \gamma(e) = c + \gamma(e)$.
3. Otherwise the next arc in P must be $(v^c, w^{c+\gamma(e)+\delta})$ for some time-edge $e = (\{v, w\}, c) \in \mathcal{E}(\mathcal{G})$. Then append to P' that time-edge e . Note that the arrival time of e is $c + \gamma'(e) \leq c + \gamma(e) + \delta$. ◀

We now show that we can use Lemma 7 to check whether s can be prevented from reaching any vertices outside of R by slowing at most k time-edges by δ each.

► **Lemma 9.** For any given $\mathcal{G} = (V, (E_i)_{i \in [\tau]}, \gamma)$, $s \in R \subseteq V$, $\delta \in \mathbb{N}$, and $k \in \mathbb{N} \cup \{0\}$, the maximum s^0 -z-flow in $\mathcal{F}(\mathcal{G}, s, R, \delta)$ has value at most k if and only if there is a set X of at most k time-edges such that s can not reach any vertices outside of R in $\mathcal{G} \uparrow_\delta X$.

Proof. Write $((V', A), c) := F := \mathcal{F}(\mathcal{G}, s, R, \delta)$.

(\Rightarrow): Let the maximum s^0 -z-flow f in F have value at most k . Moreover, let C be a s^0 -z-cut of minimum capacity. From the max-flow min-cut theorem [30], we know that $c(C) \leq k$. Note that $C \subseteq \{(e_1, e_2) \in A \mid e \in \mathcal{E}(\mathcal{G})\}$, since all other edges have infinite capacity. Hence, $|C| \leq k$. Now set $X := \{e \in \mathcal{E}(\mathcal{G}) \mid (e_1, e_2) \in C\}$. Assume towards a contradiction that there

is a temporal s - x -path P in $\mathcal{G} \uparrow_\delta X$ for some $x \in V \setminus R$. We may take P to be minimal, thus the penultimate vertex y of P is contained in R . By Lemma 8 there is a s^0 - y^t -path \hat{P} in $(V', A \setminus C)$ where t is the time P reaches y . The fact that P afterwards proceeds to x and (3) in the definition of \mathcal{F} imply that $(V', A \setminus C)$ contains a path from y^t to x . This contradicts C being a s^0 - z -cut in (V', A) .

(\Leftarrow): Let X be a time-edge set as assumed. By assumption $R_{\mathcal{G} \uparrow_\delta X}(\{s\}) \subseteq R$. We claim that $C = \{(e_1, e_2) \mid e \in X\} \subseteq A$ is a s^0 - z -cut in (V', A) , which implies that the maximum value of an s^0 - z -flow in F is at most $c(C) \leq k$ [30]. So suppose towards a contradiction that there is a s^0 - z -path P in $(V', A \setminus C)$. Let $x^t \in V$ be the penultimate vertex of P . Then there is a s - x -path P' in $\mathcal{G} \uparrow_\delta X$ with arrival time at most t by Lemma 8. The final arc of P is (x^t, z) . Hence, (x^t, z) must be contained in (3), i. e., we can extend P' by some time-edge to end at a vertex in $V \setminus R$. This contradicts our assumption $R_{\mathcal{G} \uparrow_\delta X}(\{s\}) \subseteq R$. \blacktriangleleft

If $\mathcal{F}(\mathcal{G}, s, R, \delta)$ contains a s^0 - z -flow of value $k + 1$, then we want to find a small set $Y \subseteq V(\mathcal{G}) \setminus R$ of vertices such that $Y \cap R_{\mathcal{G} \uparrow_\delta X}(s) \neq \emptyset$ for every $X \subseteq \mathcal{E}(\mathcal{G})$ with $|X| \leq k$ and $|R_{\mathcal{G} \uparrow_\delta X}(s)| \leq r$.

► Lemma 10. *Let $\mathcal{G} = (V, (E_i)_{i \in [r]}, \gamma)$, $s \in R \subseteq V$, $\delta, r \in \mathbb{N}$, and $k \in \mathbb{N} \cup \{0\}$. Assume that $\mathcal{F}(\mathcal{G}, s, R, \delta)$ has a s^0 - z -flow of value $k + 1$. We can compute in $O(k \cdot |A|)$ time a set $Y \subseteq V \setminus R$ of size at most $|R|$ such that $Y \cap R_{\mathcal{G} \uparrow_\delta X}(s) \neq \emptyset$ holds for every $X \subseteq \mathcal{E}(\mathcal{G})$ with $|X| \leq k$ and $|R_{\mathcal{G} \uparrow_\delta X}(s)| \leq r$.*

Proof. Let f be a s^0 - z -flow of value $k + 1$ in $((V', A), c) := \mathcal{F}(\mathcal{G}, s, R, \delta)$. We may assume f to never use an arc (v^t, w^t) whenever A contains some arc (v^b, z) with $b \geq t$, as we could otherwise redirect f to use that latter arc (of infinite capacity) instead. Note that performing this modification can be done in $O(k \cdot |A|)$ time. Now set

$$H := \{(v, t) \mid v^t \in V' \text{ and } (v^t, z) \in A, f((v^t, z)) > 0\}$$

By (3), we have $|H| \leq |R|$ and $N_{\mathcal{G}}(v, t) \setminus R \neq \emptyset$ for all $(v, t) \in H$. Construct the vertex set Y by picking for each $(v, t) \in H$ one arbitrary vertex from $N_{\mathcal{G}}(v, t) \setminus R$. Hence $|Y| \leq |H| \leq |R|$ and $Y \cap R = \emptyset$.

It remains to prove that $R_{\mathcal{G} \uparrow_\delta X}(s) \cap Y \neq \emptyset$, with $X \subseteq \mathcal{E}(\mathcal{G})$ being an arbitrary solution for the MINREACHSLOW-instance $(\mathcal{G}, \{s\}, k, r, \delta)$. Define $C := \{(e_1, e_2) \in A \mid e \in X\}$. Since f has value $k + 1 > c(C)$, there is a s^0 - v^t -path P in $(V', A \setminus C)$ where each edge $e \in E(P)$ has $f(e) > 0$ and $(v, t) \in H$. By Lemma 8, there is a temporal s - v -path P' in $\mathcal{G} \uparrow_\delta X$ with arrival time at most t . Note that through P' , vertex s can reach v as well as all vertices of $N_{\mathcal{G} \uparrow_\delta X}(v, t) = N_{\mathcal{G}}(v, t)$. Hence, the vertex $u \in Y$ which we picked for $(v, t) \in H$ from $N_{\mathcal{G}}(v, t) \setminus R$ is in $R_{\mathcal{G} \uparrow_\delta X}(s)$ – thus the claim is proven. \blacktriangleleft

Now we are ready to prove of Theorem 5. The corresponding search-tree algorithm is listed as Algorithm 11.

Proof of Theorem 5. Let $I = (\mathcal{G}, S, k, r, \delta)$ be the given instance of MINREACHSLOW. By Lemma 6, we can assume $S = \{s\}$ after linear time preprocessing. We now prove that Algorithm 11 solves $(\mathcal{G}, \{s\}, k, r, \delta)$ in $O(r! \cdot k \cdot |\mathcal{G}|)$ time.

Let I be a *no*-instance. Observe that line 5 ensures that at all times $|R| \leq r$. Then, by Lemma 9, $\mathcal{F}(\mathcal{G}, s, R, \delta)$ will for all $s \in R \subseteq V$ have a s^0 - z -flow of value $k + 1$. Hence, line 4, and thus Algorithm 11 will never return *yes*.

■ **Algorithm 11** Pseudocode of the algorithm behind Theorem 5.

Input: An instance $I = (\mathcal{G}, \{s\}, k, r, \delta)$ of MINREACHSLOW.
Output: *yes* if I is a *yes*-instance and otherwise *no*.

- 1 **return** $g(\{s\})$, where
- 2 **function** $g(R)$ **is**
- 3 Compute a s^0 -z-flow f in $\mathcal{F}(\mathcal{G}, s, R, \delta)$ by Lemma 7
- 4 **if** f is of value at most k **then return** *yes*
- 5 **if** $|R| \geq r$ **then return** *no*
- 6 Compute a set $Y \subseteq V \setminus R$ by Lemma 10
- 7 **foreach** $v \in Y$ **do**
- 8 **if** $g(R \cup \{v\}) = \text{yes}$ **then return** *yes*
- 9 **return** *no*

Let I be a *yes*-instance. Thus there is a set X of at most k time-edges such that $|R_{\mathcal{G}\uparrow_\delta X}(s)| \leq r$. We claim that $g(R')$ returns *yes* for all R' with $s \in R' \subseteq R_{\mathcal{G}\uparrow_\delta X}(s)$. We prove this by reverse induction on $|R'|$. In the base case where $R = R_{\mathcal{G}\uparrow_\delta X}(s)$, $g(R)$ returns *yes* by Lemma 9. Now assume the claim to hold whenever $|R| = q$ and let R be of size $q - 1$ with $s \in R \subseteq R_{\mathcal{G}\uparrow_\delta X}(s)$. Assume that $\mathcal{F}(\mathcal{G}, s, R, k, \delta)$ has a s^0 -z-flow of value $k + 1$, otherwise we are done (by line 4). By Lemma 10, the set Y computed in line 6 contains a vertex $u \in R_{\mathcal{G}\uparrow_\delta X}(s) \setminus R$. Thus, $g(R \cup \{u\})$ returns *yes* by induction hypothesis. Hence, by line 8, $g(R)$ return *yes*, completing the induction. In particular, $g(\{s\})$ returns *yes*, therefore Algorithm 11 is correct.

To bound the running time, note that each call $g(R)$ makes at most $|Y| \leq |R|$ recursive calls by Lemma 10. In each of these recursive calls the cardinality of R increases by one until $|R| = r$, so the recursion depth is at most r by line 5. Hence, we can observe by an inductive argument that the search tree has $d!$ many nodes at depth d , where the root is at depth 1. Hence, the search tree has at most $r!$ many leaves and thus $O(r!)$ many nodes in total. By Lemma 7 and Lemma 10, lines 3 and 6 take at most $O(k \cdot |\mathcal{G}|)$ time. Hence, the overall running time is bounded by $O(r! \cdot k \cdot |\mathcal{G}|)$. ◀

4 A Polynomial-Time Algorithm for Forests

In this section we present an algorithm that solves MINREACHDELETE and MINREACHDELAY in polynomial time on temporal graphs where the underlying graph is a tree or a forest. This is a quite severe yet well-motivated restriction of the input [20, 22, 21], since it could serve as the starting point for FPT-algorithms for “distance-to-forest”-parameterizations.

► **Theorem 12.** *MINREACHDELETE and MINREACHDELAY are polynomial-time solvable if the underlying graph is a forest.*

Actually we even provide an polynomial-time algorithm for a generalized version of MINREACHDELAY. Then, polynomial-time solvability of MINREACHDELETE follows from Lemma 1, since it is forest preserving. We define the generalized problem as follows:

WEIGHTED MINREACHDELAY ON FORESTS

Input: A temporal graph $\mathcal{G} = (V, (E_i)_{i \in [\tau]}, \gamma)$ whose underlying graph is a forest, a weight function $w: V \rightarrow \mathbb{N} \cup \{0, \infty\}$, a set $F \subseteq \mathcal{E}(\mathcal{G})$ of undelayable time-edges, a set of sources $S \subseteq V$, and integers k, r, δ .

Question: Does there exist a time-edge set $X \subseteq \mathcal{E}(\mathcal{G}) \setminus F$ of size at most k such that $w(R_{\mathcal{G} \nearrow_{\delta} X}(S)) \leq r$?

In the remainder of this section, we show how to solve this problem using dynamic programming in polynomial time. Informally speaking, our dynamic program works as follows. As a preprocessing step we unfold vertices of large degree, reducing to an equivalent instance of maximum degree 3. Then we root each underlying tree at an arbitrary vertex and build a dynamic programming table, starting at the leaves. More precisely, we compute a table entry for each combination of a vertex v , a budget k , a time step t , and a flag indicating whether v is first reached from a child or from its parent. This table entry then contains a minimum reachable subset of the subtree rooted at v that can be achieved by applying k delay operations to that subtree.

► **Theorem 13.** *WEIGHTED MINREACHDELAY ON FORESTS is polynomial-time solvable if the underlying graph is a forest.*

Proof. Assume for now that the underlying graph of \mathcal{G} is a tree, rooted at an arbitrary leaf. We denote by T_v the subtree with root $v \in V$. We use the reaching time ∞ to denote “never”. By convention, a vertex can reach itself at time 0. Define $\mathbb{N}^* := \mathbb{N} \cup \{0, \infty\}$.

We first show how to transform the underlying graph into a binary tree. This will highly simplify the description of the dynamic programming table. Replace each vertex v of degree $\deg(v) > 3$ by a path on $\deg(v) - 2$ new vertices, where each edge of that path is undelayable, appears at each time step and always has traversal time 0. Distribute the edges formerly incident to v among the new vertices such that each of them has degree 3. Set the weight of the path’s first vertex to the weight of v , and the weight of all other path vertices to 0. Note that this modification produces an equivalent instance of maximum degree at most 3, while increasing the number of vertices only by a constant factor.

We extend the notion of reachability to vertex-time pairs $(s, t) \in S \times [\tau]$ by saying that (s, t) reaches $v \in V$ in \mathcal{G} if there exists a temporal s - v -path starting at time t or later. For a set $A \subseteq V \times [\tau]$, $R_{\mathcal{G}}(A)$ is the set of all vertices, reachable from any member of A in \mathcal{G} . We say a vertex v is reached *through* another vertex w if there is a temporal path from a source $s \in S$ to v that uses w .

Let $v \in V$, $k \in \mathbb{N}$. Define $\mathcal{T}_{v,k}$ as the set of temporal graphs obtained from T_v by applying up to k delay operations. Partition $\mathcal{T}_{v,k}$ into $\{\mathcal{T}_{v,k,t} \mid t \in \mathbb{N}^*\}$, where $\mathcal{T}_{v,k,t}$ contains those graphs in which v is reached from $S \cap V(T_v)$ exactly at time t . Finally, we set for each $t \in \mathbb{N}^*$

$$D[v, k, t, \text{false}] = \min \{w(R_T(S \cap V(T_v))) \mid T \in \mathcal{T}_{v,k,t}\} \quad \text{and}$$

$$D[v, k, t, \text{true}] = \min \left\{ w(R_T(S \cap V(T_v)) \cup R_T(\{(v, t)\})) \mid T \in \bigcup_{t' \geq t} \mathcal{T}_{v,k,t'} \right\}.$$

where the minimum of an empty set is ∞ by convention. It is convenient to also define these entries as ∞ whenever $k < 0$. Roughly speaking, $D[v, k, t, \iota]$ contains the minimal weight reached in T_v under the assumption that up to k delay operations are applied to T_v , that v is first reached at time t , and that

- v is reached by a source in $S \cap V(T_v)$ at time t if $\iota = \text{false}$,
- v is reached by a source in $S \setminus V(T_v)$ at time t if $\iota = \text{true}$.

Note that v might be reached simultaneously from $S \cap V(T_v)$ and $S \setminus V(T_v)$.

76:12 Temporal Reachability Minimization: Delaying vs. Deleting

We next show how to compute $D[v, k, t, \iota]$ recursively, starting at the leaf vertices. Observe that $D[v, k, t, \iota] = \infty$ whenever $v \in S$ is a source and $t > 0$. Thus, this case shall be excluded in the following.

If v has no children. If $\iota = \mathbf{false}$ and $v \notin S$ and $t < \infty$, then $D[v, k, t, \mathbf{false}] = \infty$ as there is no way that v can be reached from a source in $S \cap V(T_v) = \emptyset$. Otherwise,

$$D[v, k, t, \iota] = w(v) \cdot [t < \infty].$$

If v has exactly one child v' . If $\iota = \mathbf{false}$ and $v \notin S$, then v must be reached through v' at time t . In this case the minimal total weight reached in $T_{v'}$ is

$$D_1 := \min_{t' \leq t} D[v', k - \kappa(t', t), t', \mathbf{false}],$$

where $\kappa(t', t)$ is the minimal number of delays that need to occur on the edge $\{v, v'\}$ to ensure that (v', t') reaches v at time t but not earlier. (Set $\kappa(t', t) = \infty$ if this is impossible.) Consequently, if $v \notin S$, then

$$D[v, k, t, \mathbf{false}] = w(v) \cdot [t < \infty] + D_1.$$

If $\iota = \mathbf{true}$ or $v \in S$, then there are two possibilities. If v' is reached through v at time t' , with t' being the first time v' is reached from S , then the minimal total weight reached in $T_{v'}$ is

$$D_2 := \min_{t' \geq t} D[v', k - \kappa(t, t'), t', \mathbf{true}].$$

Otherwise, v' must be reached from a source in $S \cap V(T_{v'})$ at time t' , thus the minimal total weight reached in $T_{v'}$ is

$$D_3 := \min_{t'} D[v', k - \hat{\kappa}(t', t), t', \mathbf{false}],$$

where $\hat{\kappa}(t', t)$ is the minimal number of delays that need to occur on $\{v, v'\}$ to ensure that (v, t) cannot reach v' before time t' and (v', t') cannot reach v before time t . (Again, set $\hat{\kappa}(t', t) = \infty$ if this is impossible.) Thus we obtain for the case that $\iota = \mathbf{true}$ or $v \in S$ that

$$D[v, k, t, \iota] = w(v) \cdot [t < \infty] + \min\{D_2, D_3\}.$$

If v has two children v', v'' . The situation is similar to that of only one child vertex, although more possible cases have to be distinguished. We omit the tedious details. However, it is clear that $D[v, k, t, \iota]$ can be computed by simply trying all possible tuples $(t', t'', k', k'', i', i'', l', l'')$ where t', t'' are the times at which v', v'' are reached; k', k'' are the number of delays occurring in $T_{v'}, T_{v''}$; i', i'' are the number of delays occurring on the edges $\{v, v'\}, \{v, v''\}$; and l', l'' describe whether v', v'' are reached from a source in their respective subtrees at time t' and t'' , respectively. The number of such tuples and the time required to process each of them is clearly polynomial in $t + k + |\mathcal{G}|$.

After having computed all entries $D[v, k, t, \iota]$, the solution of the MINREACHDELAY instance (\mathcal{G}, k) can be found as the value of

$$R[\hat{v}, k] := \min_t D[\hat{v}, k, t, \mathbf{false}],$$

where \hat{v} is the root vertex of \mathcal{G} .

It remains to consider the case that the underlying graph of \mathcal{G} is a disconnected forest. In this case simply apply the above algorithm to each connected component. Afterwards, determining the optimal way to split the overall budget between the connected components can be computed by a simple dynamic program. Define $X[i, k]$ as the minimum weight reached in the first i trees if up to k time-edges are delayed and use the fact that

$$X[1, k] = \min_{k' \leq k} (R[\hat{v}_1, k'])$$

and for all $i > 1$

$$X[i, k] = \min_{k' \leq k} (R[\hat{v}_i, k'] + X[i-1, k-k']),$$

where \hat{v}_j is the root of the j^{th} tree. ◀

5 Conclusion

While both problem variants, MINREACHDELETE and MINREACHDELAY, are polynomial-time solvable on forests and W[1]-hard when parameterized by k , even if the lifetime is $\tau = 2$, their complexities diverge when we parameterize by the number r of reachable vertices. Here, MINREACHDELETE is W[1]-hard while for MINREACHDELAY we found a fixed-parameter tractable algorithm. This makes MINREACHDELAY particularly interesting for applications where the number of reachable vertices should be very small, e.g. when trying to contain the spread of dangerous diseases.

On the practical side we want to point out that our algorithm for MINREACHDELAY parameterized by r uses only linear space, and its search-tree-based approach makes it fit for optimization techniques like further data reduction rules or pruning using lower bounds. Furthermore, our max-flow-based branching technique can be turned into a r -approximation for minimizing the number r of reachable vertices by delaying k time-edges. To do so, instead of branching into all choices of $v \in Y$ in line 8 of Algorithm 11, simply invoke $g(R \cup Y)$. Refining the presented technique towards better approximation guarantees seems to be a promising research direction. Moreover, when focusing on specific applications, it is natural to exploit application-dependent graph properties towards designing more efficient algorithms. In particular: which well-motivated temporal graph classes beyond trees allow e.g. polynomial-time solvability of MINREACHDELETE or MINREACHDELAY? Finally, from the viewpoint of parameterized complexity the parameters k and r are settled, but the landscape of structural parameters is still waiting to be explored.

References

- 1 Infection prevention and control and preparedness for covid-19 in healthcare settings, 2020. URL: <https://www.ecdc.europa.eu/en/publications-data/infection-prevention-and-control-and-preparedness-covid-19-healthcare-settings>.
- 2 Eleni C. Akrida, George B. Mertzios, Sotiris E. Nikolettseas, Christoforos L. Raptopoulos, Paul G. Spirakis, and Viktor Zamaraev. How fast can we reach a target vertex in stochastic temporal graphs? *Journal of Computer and System Sciences*, 114:65–83, 2020. doi:10.1016/j.jcss.2020.05.005.
- 3 Eleni C. Akrida, George B. Mertzios, Paul G. Spirakis, and Christoforos L. Raptopoulos. The temporal explorer who returns to the base. *Journal of Computer and System Sciences*, 120:179–193, 2021. doi:10.1016/j.jcss.2021.04.001.
- 4 Kyriakos Axiotis and Dimitris Fotakis. On the size and the approximability of minimum temporally connected subgraphs. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 149:1–149:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.149.

- 5 Matthias Bentert, Anne-Sophie Himmel, André Nichterlein, and Rolf Niedermeier. Efficient computation of optimal temporal walks under waiting-time constraints. *Applied Network Science*, 5(1):73, 2020. doi:10.1007/s41109-020-00311-0.
- 6 Kenneth A Berman. Vulnerability of scheduled networks and a generalization of menger's theorem. *Networks*, 28(3):125–134, 1996. doi:10.1002/(SICI)1097-0037(199610)28:3<125::AID-NET1>3.0.CO;2-P.
- 7 Hans L. Bodlaender and Tom C. van der Zanden. On exploring always-connected temporal graphs of small pathwidth. *Information Processing Letters*, 142:68–71, 2019. doi:10.1016/j.ipl.2018.10.016.
- 8 Dirk Brockmann and Dirk Helbing. The hidden geometry of complex, network-driven contagion phenomena. *Science*, 342(6164):1337–1342, 2013. doi:10.1126/science.1245200.
- 9 Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003. doi:10.1142/S0129054103001728.
- 10 Sebastian Buß, Hendrik Molter, Rolf Niedermeier, and Maciej Rymar. Algorithmic aspects of temporal betweenness. In *Proceedings of the 26th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2084–2092, 2020. doi:10.1145/3394486.3403259.
- 11 Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. In *Proceedings of the 31st International Symposium on Algorithms and Computation (ISAAC)*, volume 181, pages 30:1–30:18, 2020. To appear in *Algorithmica*. doi:10.4230/LIPIcs.ISAAC.2020.30.
- 12 Arnaud Casteigts, Joseph G. Peters, and Jason Schoeters. Temporal cliques admit sparse spanners. *Journal of Computer and System Sciences*, 121:1–17, 2021. doi:10.1016/j.jcss.2021.04.004.
- 13 Vittoria Colizza, Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences of the United States of America*, 103(7):2015–2020, 2006. doi:10.1073/pnas.0510525103.
- 14 Daryl J Daley and David G Kendall. Epidemics and rumours. *Nature*, 204(4963):1118–1118, 1964. doi:10.1038/2041118a0.
- 15 Argyrios Deligkas and Igor Potapov. Optimizing reachability sets in temporal graphs by delaying. In *Proceedings of the 34th Conference on Artificial Intelligence (AAAI)*, pages 9810–9817, 2020. doi:10.1609/aaai.v34i06.6533.
- 16 Reinhard Diestel. *Graph Theory*, volume 173. Springer, 5 edition, 2016. doi:10.1007/978-3-662-53622-3.
- 17 Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. doi:10.1007/BF01386390.
- 18 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
- 19 Ken TD Eames and Matt J Keeling. Contact tracing and disease control. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1533):2565–2571, 2003. doi:10.1098/rspb.2003.2554.
- 20 Jessica Enright and Kitty Meeks. Deleting edges to restrict the size of an epidemic: a new application for treewidth. *Algorithmica*, 80(6):1857–1889, 2018. doi:10.1007/s00453-017-0311-7.
- 21 Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021. doi:10.1016/j.jcss.2021.01.007.
- 22 Jessica Enright, Kitty Meeks, and Fiona Skerman. Assigning times to minimise reachability in temporal graphs. *Journal of Computer and System Sciences*, 115:169–186, 2021. doi:10.1016/j.jcss.2020.08.001.
- 23 Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. *Journal of Computer and System Sciences*, 119:1–18, 2021. doi:10.1016/j.jcss.2021.01.005.

- 24 Thomas Erlebach, Frank Kammer, Kelin Luo, Andrej Sajenko, and Jakob T. Spooner. Two moves per time step make a difference. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 141:1–141:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.141.
- 25 Thomas Erlebach and Jakob T. Spooner. Faster exploration of degree-bounded temporal graphs. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 117, pages 36:1–36:13, 2018. doi:10.4230/LIPIcs.MFCS.2018.36.
- 26 Thomas Erlebach and Jakob T. Spooner. Non-strict temporal exploration. In *Proceedings of the 27th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 12156, pages 129–145, 2020. doi:10.1007/978-3-030-54921-3_8.
- 27 Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science*, 2020. doi:10.1126/science.abb6936.
- 28 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020. doi:10.1016/j.tcs.2019.03.031.
- 29 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.
- 30 L. R. Ford, Jr. and D. R. Fulkerson. *Flows in networks*. Princeton University Press, 1962. doi:10.1515/9781400875184.
- 31 William Goffman and V Newill. Generalization of epidemic theory. *Nature*, 204(4955):225–228, 1964. doi:10.1038/204225a0.
- 32 Roman Haag, Hendrik Molter, Rolf Niedermeier, and Malte Renken. Feedback edge sets in temporal graphs. In *Proceedings of the 46th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 12301, pages 200–212, 2020. doi:10.1007/978-3-030-60440-0_16.
- 33 John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973. doi:10.1145/362248.362272.
- 34 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. doi:10.1006/jcss.2002.1829.
- 35 George B Mertzios, Othon Michail, and Paul G Spirakis. Temporal network optimization subject to connectivity constraints. *Algorithmica*, 81(4):1416–1449, 2019. doi:10.1007/s00453-018-0478-6.
- 36 Andrew Mitchell, David Bourn, J Mawdsley, William Wint, Richard Clifton-Hadley, and Marius Gilbert. Characteristics of cattle movements in britain—an analysis of records from the cattle tracing system. *Animal Science*, 80(3):265–273, 2005. doi:10.1079/ASC50020265.
- 37 Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(14):3200, 2001. doi:10.1103/PhysRevLett.86.3200.
- 38 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):1–24, 2008. doi:10.1145/1391289.1391291.
- 39 Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.
- 40 Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. Efficient algorithms for temporal path computation. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2927–2942, 2016. doi:10.1109/TKDE.2016.2594065.
- 41 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding small separators in temporal graphs. *Journal of Computer and System Sciences*, 107:72–92, 2020. doi:10.1016/j.jcss.2019.07.006.

A Timecop’s Chase Around the Table

Nils Morawietz 

Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany

Petra Wolf  

Fachbereich 4 Informatikwissenschaften, Universität Trier, Germany

Abstract

We consider the cops and robbers game variant consisting of one cop and one robber on time-varying graphs (TVG). The considered TVGs are edge periodic graphs, i.e., for each edge, a binary string s_e determines in which time step the edge is present, namely the edge e is present in time step t if and only if the string s_e contains a 1 at position $t \bmod |s_e|$. This periodicity allows for a compact representation of an infinite TVG. We prove that even for very simple underlying graphs, i.e., directed and undirected cycles the problem whether a cop-winning strategy exists is NP-hard and W[1]-hard parameterized by the number of vertices. Our second main result are matching lower bounds for the ratio between the length of the underlying cycle and the least common multiple (lcm) of the lengths of binary strings describing edge-periodicities over which the graph is robber-winning. Our third main result improves the previously known EXPTIME upper bound for PERIODIC COP & ROBBER on general edge periodic graphs to PSPACE-membership.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Design and analysis of algorithms

Keywords and phrases Time variable graph, Edge periodic cycle, Game of cops and robbers, Computational complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.77

Related Version Full Version: <https://arxiv.org/abs/2104.08616>

Funding Nils Morawietz: Supported by Deutsche Forschungsgemeinschaft (DFG), project OPERAH, KO 3669/5-1.

Petra Wolf: Supported by Deutsche Forschungsgemeinschaft (DFG), project FE560/9-1.

1 Introduction

In general, a *time-varying graph* (TVG) describes a graph that varies over time. For most applications, this variation is limited to the availability or weight of edges meaning that edges are only present at certain time-steps or the time needed to cross an edge changes over time. TVGs are of great interest in the area of *dynamic networks* [4, 9, 10, 11] such as *mobile ad hoc networks* [19] and *vehicular networks* modeling traffic load factors on a road network [7]. In those networks, the topology naturally changes over time and TVGs are used to reflect this dynamic behavior. Quite recently, TVGs became of interest in the context of graph games such as competitive diffusion games and Voronoi games [2]. There are plenty of representations for TVGs in the literature which are not equivalent in general. For instance, in [4] a TVG is defined as a tuple $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ where V is a set of vertices, $E \subseteq V \times V \times L$ is a set of labeled edges (with labels from a set L), $\mathcal{T} \subseteq \mathbb{T}$ is the *lifetime* of the graph, \mathbb{T} is the temporal domain and assumed to be \mathbb{N} for discrete systems and \mathbb{R}^+ for continuous systems, $\rho: E \times \mathcal{T} \rightarrow \{0, 1\}$ is the *presence function* indicating whether an edge e is present in time step t , and $\zeta: E \times \mathcal{T} \rightarrow \mathbb{T}$ is the latency function indicating the time needed to cross edge e in time step t . We call the graph $G = (V, E)$ the *underlying graph* of \mathcal{G} . The literature has not yet agreed on how the function ρ (and ζ) are given in the input. This is of significant importance, if ρ exhibits periodicity with respect to single edges in the context of computational complexity. We say that a TVG belongs to the class of TVGs featuring *periodicity of edges*, defined as class 8 in [4], if $\forall e \in E, \forall t \in \mathcal{T}, \forall k \in \mathbb{N}, \rho(e, t) = \rho(e, t + kp_e)$



© Nils Morawietz and Petra Wolf;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 77; pp. 77:1–77:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for some $p_e \in \mathbb{T}$, depending on e , and the underlying graph G is connected. For these TVGs, the function ρ can be represented for each edge $e \in E$ as a binary string of size p_e concatenating the values of $\rho(e, t)$ for $0 \leq t < p_e$. Note that the period of the whole graph \mathcal{G} is then the least common multiple (lcm for short) of all string lengths p_e describing edge periods. Therefore, the underlying graph G of \mathcal{G} can have exponentially many different sub-graphs G_t representing a snapshot of \mathcal{G} at time t . This exponential blow-up is a huge challenge in determining the precise complexity of problems for TVGs featuring periodicity of edges as discussed in more detail in Section 5 and 6. Often, for general TVGs a representation containing all sub-graphs representing snapshots over the whole lifetime of the graph is chosen when the complexity of decision problems over TVGs are considered [1, 13]. An approach to unify the representation of TVGs is given in [18], also including the existence of vertices being effected over time. This approach represents $\rho(e, t)$ by enhancing an edge $e = (u, v)$ with the departure time t_d at u and the arrival time t_a at v , where t_a might be smaller than t_d in order to model periodicity. As for TVGs with periodicity of edges where ρ is represented as a binary string for each edge the periodicity of the TVG \mathcal{G} might be exponential in its representation using the approach in [18] would also cause an exponential blow-up in the representation of \mathcal{G} , as a decrement of the time value could only be used after a whole period of the graph, rather than after the period of one edge. An other class of TVGs based on periodicity was considered in the field of robotics to model motion planning tasks if time dependent obstacles are present [17]. There, the availability of the vertices in the graph changes periodically and each edge needs a constant number of time steps to be crossed. An edge $e = \{u, v\}$ is only present if in the time span needed to cross e both endpoints are u and v are still present. In [17] the periodic function describing the availability of a vertex and the function describing the time needed to cross an edge is represented by an on-line program and can hence handle values exponential in their representation. This is crucial in the PSPACE-hardness proof of the reachability problem for graphs in this class presented in [17]. There, the hardness is obtained by a reduction from the halting problem for linear space-bounded deterministic Turing machines where a configuration of the Turing machine is encoded in the time step. In the reduction, the periodicity of a single vertex as well as the time needed to cross an edge is of value exponential in the tape length-bound. Note that this representation of periodicity is exponentially more compact than in our setting and thus the result of [17] does not translate to our setting.

We will stick in the following to the model describing TVG featuring periodicity of edges where the function $\rho(e, t)$ is represented as a binary string. For this representation, Erlebach and Spooner [8] introduced recently a variant of the famous cops and robber problem which is intensively studied for static graphs [3]. In the static setting, the game is played on a given graph and includes a single robber and a set of k -cops. The cops and robber occupy vertices and the cops choose their vertices first. Then, in each round the players alternate turns and the cops move first. Thereby, each cop can move to an adjacent vertex or pass and stay on her vertex. The same holds for the robber. We say that a graph is *k-cop-winning*, if there exists a strategy for the k cops in which they finally catch the robber, i.e., a cop occupies the same vertex as the robber. If the context is clear, we call a 1-cop-winning graph a *cop-winning* graph. If a graph is not cop-winning, we call it *robber-winning*. A special interest on the game of cops and robbers lied in characterizing graphs which are *k-cop-winning*. While for one cop, the cop-winning graphs where understood in 1978 and independently 1983 [16, 15] as those featuring a special kind of ordering on the vertex set, called a *cop-win* or *eliminating* order, the case for k cops was long open and solved in 2009 by exploiting a linear structure of a certain power of the graph [5].

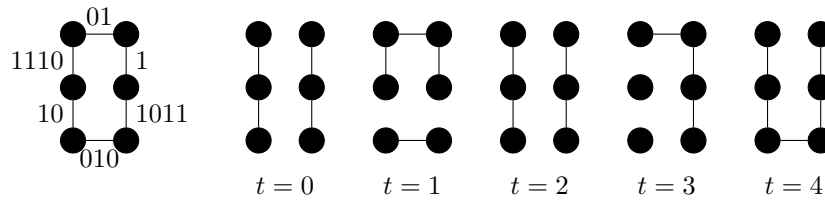
In 2020 Erlebach and Spooner [8] connected the two discussed graph-theoretical topics of great interest by introducing a cops and robber game for *edge periodic graphs*. These are TVGs featuring periodicity of edges as discussed above. They gave an algorithm which determines if the given edge periodic graph is k -cop-winning, which runs in time $\mathcal{O}(\text{lcm} \cdot k \cdot |G|^{k+2})$, where lcm is the least common multiple of all length of binary strings describing $\rho(e, t)$ and G is the underlying graph. As lcm can be exponentially in the input size, they proposed the question of whether this problem is NP-hard. This question was answered positively for a one-cop version, also in 2020, by Morawietz, Rehs, and Weller [14] for TVGs of which the underlying graph has a constant size vertex cover or where two edges have to be removed to obtain a cycle. Moreover, they showed that the problem is W[1]-hard when parameterized by the size of the underlying graph G even in these restricted cases, that is, they showed that there is presumably no algorithm solving the problem in time $f(|G|) \cdot n^{\mathcal{O}(1)}$ for any computable function f . In other words, the exponential growth of the running time of every algorithm solving the problem has to depend on the length of binary strings describing $\rho(e, t)$.

Our contribution. In this work, we show, that the NP-hardness holds for even simpler classes of edge periodic graphs, namely for directed and undirected cycles. Moreover, we show that the W[1]-hardness when parameterized by $|G|$ even holds for these restricted instances (Section 3). The quite restricted class of undirected cycles was also studied in [8] where an upper bound on the length of the cycle with respect to the lcm was given for which each edge periodic cycle is robber winning. To be more precise, for an edge periodic cycle on n vertices it holds that if $n \geq 2 \cdot \ell \cdot \text{lcm}$, then the graph is robber winning. Here, $\ell = 1$ if lcm is at least two times the longest size of a binary string describing $\rho(e, t)$ and $\ell = 2$, otherwise. For these upper bounds only non-matching lower bounds were given, i.e., a family of cop-winning edge periodic cycles with length $3 \cdot \text{lcm}$ for $\ell = 2$ and $1.5 \cdot \text{lcm}$ for $\ell = 1$ are given. These lower bounds left a gap of size $0.5 \cdot \ell \cdot \text{lcm}$. In this work, we show that the given upper bounds are tight by closing this gap by giving families of cop-winning edge periodic cycles of length $2 \cdot \ell \cdot \text{lcm} - 1$ (Section 4). Finally, we improve the currently best EXPTIME upper bound shown in [8] for the problem, whether a given edge periodic graph is cop-winning to PSPACE (Section 5). We conclude with a discussion on the restricted class of directed edge periodic cycles indicating that due to the compact representation of the edge periodic graphs, which does not introduce additional amounts of freedom, the standard complexity classes, such as NP and PSPACE might not be suitable to precisely characterize the complexity of this problem (Section 6).

2 Preliminaries

For a string $w = w_0 w_1 \dots w_n$ with $w_i \in \{0, 1\}$, for $0 \leq i \leq n$, we denote with $w[i]$ the symbol w_i at position i in w . We write the concatenation of strings u and v as $u \cdot v$. For non-negative integers $i \leq j$ we denote with $[i, j]$ the interval of natural numbers n with $i \leq n \leq j$.

An *edge periodic (temporal) graph* $\mathcal{G} = (V, E, \tau)$ (see also [8]) consists of a graph $G = (V, E)$ (called the *underlying graph*) and a function $\tau : E \rightarrow \{0, 1\}^*$ where τ maps each edge e to a sequence $\tau(e) \in \{0, 1\}^*$ such that e exists in a time step $t \geq 0$ if and only if $\tau(e)[t]^\circ = 1$, where $\tau(e)[t]^\circ := \tau(e)[t \bmod |\tau(e)|]$. For an edge e and non-negative integers $i \leq j$ we inductively define $\tau(e)[[i, j]]^\circ = \tau(e)[i]^\circ \cdot \tau(e)[[i+1, j]]^\circ$ and $\tau(e)[[j, j]]^\circ = \tau(e)[j]^\circ$. If $\tau(e) = 1$, we call e a *1-edge*. Every edge e exists in at least one time step, that is, for each edge e there is some $t_e \in [0, |\tau(e)| - 1]$ with $\tau(e)[t_e] = 1$. We might abbreviate i repetitions of the same symbol σ in $\tau(e)$ as σ^i . Let $L_{\mathcal{G}} = \{|\tau(e)| \mid e \in E\}$ be the set of all edge periods



■ **Figure 1** Edge periodic cycle \mathcal{G} (left) together with snapshots $\mathcal{G}(t)$ for $0 \leq t \leq 4$.

of some edge periodic graph $\mathcal{G} = (V, E, \tau)$ and let $\text{lcm}(L_{\mathcal{G}})$ be the least common multiple of all periods in $L_{\mathcal{G}}$. We call an edge periodic graph \mathcal{G} with an underlying graph consisting of a single cycle an *edge periodic cycle*. We denote with $\mathcal{G}(t)$ the sub-graph of G present in time step t . We do not assume that \mathcal{G} is connected in any time step. We will discuss directed and undirected edge periodic graphs. If not stated otherwise, we assume an edge periodic graph to be undirected. We illustrate the notion of edge periodic cycles in Figure 1 showing an edge periodic cycle \mathcal{G} together with $\mathcal{G}(t)$ for the first 5 time steps. We now define the considered cops and robbers variant on edge periodic graphs with one single cop. Here, first the cop chooses her start vertex in $\mathcal{G}(0)$, then the robber chooses his start vertex in $\mathcal{G}(0)$. Then, in each time step t , the cop and robber move to an adjacent vertex over an edge which is present in $\mathcal{G}(t)$ or pass and stay on their vertex. Thereby, the cop moves first, followed by the robber. We say that the cop catches the robber, if there is some time step t for which the cop and the robber are on the same vertex after the cop moved. If the cop has a strategy to catch the robber regardless which start vertex the robber chooses, we say that \mathcal{G} is *cop-winning* and call the strategy implemented by the cop a *cop-winning strategy*. If for all cop start vertices, there exists a start vertex and strategy for the robber to elude the cop indefinitely, we call \mathcal{G} *robber-winning*. The described game is a zero-sum game, i.e., \mathcal{G} is either cop-winning or robber-winning.

PERIODIC COP & ROBBER

Input: An edge periodic graph $\mathcal{G} = (V, E, \tau)$.

Question: Is \mathcal{G} cop-winning?

3 It's hard to run around a table

In this section, we show that the NP-hardness of PERIODIC COP & ROBBER already holds if the input graphs are very restricted. More precisely, we show that PERIODIC COP & ROBBER is NP-hard and W[1]-hard when parameterized by $|G|$, even for directed and undirected edge periodic cycles \mathcal{G} .

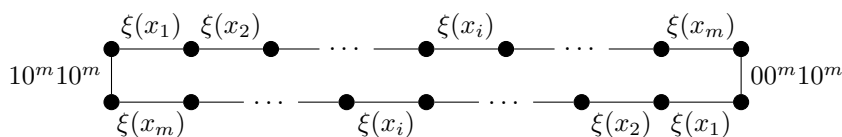
► **Theorem 1.** *PERIODIC COP & ROBBER on directed or undirected edge periodic cycles is NP-hard, and W[1]-hard parameterized by the size of the underlying graph G .*

Both, the undirected and directed case of Theorem 1 is shown by a reduction of the PERIODIC CHARACTER ALIGNMENT problem which was shown to be both NP-hard and W[1]-hard when parameterized by $|X|$ in [14].

PERIODIC CHARACTER ALIGNMENT

Input: A finite set $X \subseteq \{0, 1\}^*$ of binary strings.

Question: Is there a position i , such that $x[i]^\circ = 1$ for all $x \in X$, where $x[i]^\circ := x[i \bmod |x|]$?



■ **Figure 2** PERIODIC COP & ROBBER instance constructed from a PERIODIC CHARACTER ALIGNMENT instance with set of strings $X = \{x_1, \dots, x_m\}$ in the proof of Lemma 2. For $x_j \in X$ the edge labels are defined as $\xi(x_j) := \xi(x_j[0]) \cdot \xi(x_j[1]) \cdot \dots \cdot \xi(x_j[|x_j| - 1])$, with $\xi(c) := 0c^m 01^m$ for $c \in \{0, 1\}$. The upper chain corresponds to the vertices r_j and the lower chain to the vertices ℓ_j .

We begin with considering the case of undirected edge periodic cycles and then proceed by adapting the obtained construction for directed edge periodic cycles.

► **Lemma 2.** *PERIODIC COP & ROBBER on undirected edge periodic cycles is NP-hard and W[1]-hard parameterized by the size of the underlying graph G .*

Proof. We first sketch the idea of the construction. It is helpful to consider Figure 2 in the following. We represent each string in X by an edge label. The constructed cycle will consist of two chains connected by two special edges. In the first chain, the elements in X are increasingly listed in some fixed order as individual edge labels each. In the second chain the same edge labels are listed decreasingly in the same order. This will allow the cop and the robber to occupy antipolar vertices with the same edge labels on incident edges. Hence, while the cop is on one chain and the robber on the other chain, the robber can mimic the movements of the cop. The two chains are connected by two special edges for which their edge labels are complementary in one position of the labels and identical in all other positions. This will allow the cop to switch between the chains in a certain time step while the robber is trapped on his chain. In this situation, the cop will be able to catch the robber if and only if there is a position i , such that $x[i]^\circ = 1$ for all $x \in X$, in which case all edges of the chains will be present for some period.

We now proceed with the formal proof. Let X be an instance of PERIODIC CHARACTER ALIGNMENT. We describe how to construct in polynomial time an instance $\mathcal{G} = (V, E, \tau)$ of PERIODIC COP & ROBBER, where \mathcal{G} is an undirected edge periodic cycle such that X is a yes-instance of PERIODIC CHARACTER ALIGNMENT if and only if \mathcal{G} is a yes-instance of PERIODIC COP & ROBBER.

Let $|X| = m$ and $\{x_1, \dots, x_m\}$ be the elements of X . We set $V := \{\ell_j, r_j \mid 0 \leq j \leq m\}$ and $E := \{\{\ell_{j-1}, \ell_j\}, \{r_{j-1}, r_j\} \mid 1 \leq j \leq m\} \cup \{\{\ell_0, r_m\}, \{\ell_m, r_0\}\}$. Next, we set $\tau(\{\ell_0, r_m\}) := 10^m 10^m$ and $\tau(\{\ell_m, r_0\}) := 00^m 10^m$. Let $\xi(c) := 0c^m 01^m$ for all $c \in \{0, 1\}$. Finally, we set $\tau(\{\ell_{j-1}, \ell_j\}) := \tau(\{r_{j-1}, r_j\}) := \xi(x_j[0]) \cdot \xi(x_j[1]) \cdot \dots \cdot \xi(x_j[|x_j| - 1])$ for each $x_j \in X$. Note that the length of each edge label is divisible by $q := 2m + 2$. For $i \geq 0$, let $T_i := [q \cdot i, q \cdot (i + 1) - 1]$ denote the i -th time block, that is, the q consecutive time steps starting from $q \cdot i$. Note that the j -th edge label limited to the i -th time block $\tau(\{\ell_{j-1}, \ell_j\})[T_i]^\circ = \tau(\{r_{j-1}, r_j\})[T_i]^\circ$ is exactly $\xi(x_j[i]^\circ)$.

Next, we show that X is a yes-instance of PERIODIC CHARACTER ALIGNMENT if and only if \mathcal{G} is a yes-instance of PERIODIC COP & ROBBER.

(\Rightarrow) Let i be a position such that $x[i]^\circ = 1$ for all $x \in X$. We describe the winning strategy for the cop. She should choose the vertex ℓ_0 as her start vertex and should never move until the beginning t of the i -th time block T_i . Since $x[i]^\circ = 1$ for all $x \in X$, $\tau(\{\ell_{j-1}, \ell_j\})[T_i]^\circ = \tau(\{r_{j-1}, r_j\})[T_i]^\circ = \xi(1) = 01^m 01^m$. Consequently, in time step t only the edge $\{\ell_0, r_m\}$ exists and in the following m time steps, all edges except $\{\ell_0, r_m\}$ and $\{\ell_m, r_0\}$ exist.

If the robber is currently on some vertex r_j , then the cop should move to r_m in time step t . Otherwise, the cop should stay on ℓ_0 in this time step. By the fact that the edge $\{\ell_m, r_0\}$ does not exist in time step t , we obtain that, at the beginning of time step $t + 1$, both players are either on vertices labeled with r or labeled with ℓ . Since all edges of the two paths (ℓ_0, \dots, ℓ_m) and (r_0, \dots, r_m) exist in the time steps $[t + 1, t + m]$, the cop can catch the robber in at most m time steps, since neither $\{\ell_0, r_m\}$ nor $\{\ell_m, r_0\}$ exists in any of the time steps $[t + 1, t + m]$. Consequently, \mathcal{G} is a yes-instance of PERIODIC COP & ROBBER.

(\Leftarrow) Suppose that X is a no-instance of PERIODIC CHARACTER ALIGNMENT. We describe a winning strategy for the robber. In the following, we say that the vertex ℓ_j is the *mirror vertex* of r_j and vice versa. Moreover, we say that the robber *mirrors the move* of the cop at some time step t , if the cop is on the mirror vertex of the robber at the beginning of time step t and the robber moves to the mirror vertex of the vertex the cop ends on in time step t .

The start vertex of the robber should be the mirror vertex of the start vertex of the cop. If it is possible, then the robber should always mirror the moves of the cop.

Note that the only move the robber *cannot* mirror is if the cop traverses the edge $\{\ell_m, r_0\}$ at the beginning of some i -th time block.

We show that the robber has a strategy to end on the mirror vertex during the i -th time block and evade the cop until then.

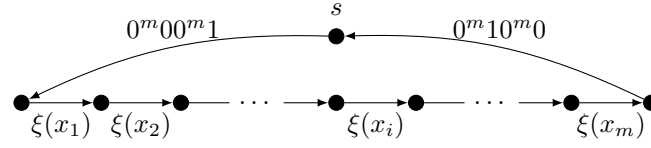
Assume w.l.o.g., that the cop moves from ℓ_m to r_0 and, thus, the robber is currently on r_m . Since X is a no-instance of PERIODIC CHARACTER ALIGNMENT, there is at least one $x_j \in X$ with $x_j[i]^\circ = 0$. Hence, $\tau(\{\ell_{j-1}, \ell_j\})[T_i]^\circ = \tau(\{r_{j-1}, r_j\})[T_i]^\circ = \xi(0) = 00^m 01^m$. Consequently, the cop cannot catch the robber in the first $m + 1$ time steps of the i -th time block. Hence, the robber should stay on this vertex until the beginning of time step $q \cdot i + m + 1$.

If the cop moves from r_0 to ℓ_m in time step $q \cdot i + m + 1$, the robber is again on the mirror vertex of the cop and is able to mirror all of the cop's moves in the remaining time steps of this time block. Otherwise, the cop stays on some vertex r_p . In this case, the robber should move to ℓ_0 . Since the edges $\{\ell_0, r_m\}$ and $\{\ell_m, r_0\}$ do not exist in the remaining time steps of this time block, the cop cannot catch the robber in this time block. Moreover, since all edges of the path (ℓ_0, \dots, ℓ_m) exist in the last m time steps of the i -th time block, the robber can move along the path (ℓ_0, \dots, ℓ_m) and reach the mirror vertex of the cop in at most m time steps. Consequently, we can show via induction, that the robber has an infinite evasive strategy and, thus, \mathcal{G} is a no-instance of PERIODIC COP & ROBBER. Since PERIODIC CHARACTER ALIGNMENT is W[1]-hard when parameterized by $|X|$ and $|V| = |E| = 2 \cdot |X| + 2$, we obtain that PERIODIC COP & ROBBER is W[1]-hard when parameterized by the size of the underlying graph of \mathcal{G} even on undirected edge periodic cycles. \blacktriangleleft

Next, we adapt the previous construction for *directed* edge periodic cycles. It is helpful to consider Figure 3 in the following. In the adaption, we only have one chain listing the elements of X . The end vertex of this chain is connected to a new vertex s which is again connected to the start vertex of the chain. The edges incident with s will act as the two edges connecting the two chains in the previous construction by delaying the robber, such that the cop can catch him if all edges corresponding to X are present in some time period.

► **Lemma 3.** *PERIODIC COP & ROBBER on directed edge periodic cycles is NP-hard, and W[1]-hard parameterized by the size of the underlying graph.*

Proof. Again, we reduce from PERIODIC CHARACTER ALIGNMENT. Let X be an instance of PERIODIC CHARACTER ALIGNMENT. We describe how to construct an instance $\mathcal{G} = (V, E, \tau)$ of PERIODIC COP & ROBBER, where \mathcal{G} is a directed edge periodic cycle. Let $|X| = m$ and $\{x_1, \dots, x_m\}$ be the elements of X . We set $V := \{v_j \mid$



■ **Figure 3** PERIODIC COP & ROBBER instance constructed from a PERIODIC CHARACTER ALIGNMENT instance with set of strings $X = \{x_1, \dots, x_m\}$ in the proof of Lemma 3. For $x_j \in X$ the edge labels are defined by the homomorphism $\xi(x_j) := \xi(x_j[0]) \cdot \xi(x_j[1]) \cdot \dots \cdot \xi(x_j[|x_j| - 1])$, with $\xi(c) := c^m 01^m 1$ for $c \in \{0, 1\}$.

$0 \leq j \leq m\} \cup \{s\}$ and $E := \{(v_{j-1}, v_j) \mid 1 \leq j \leq m\} \cup \{(v_m, s), (s, v_0)\}$. Next, we set $\tau((v_m, s)) := 0^m 10^m 0$ and $\tau((s, v_0)) := 0^m 00^m 1$. Let $\xi(c) := c^m 01^m 1$ for all $c \in \{0, 1\}$. Finally, we set $\tau((v_{j-1}, v_j)) := \xi(x_j[0]) \cdot \xi(x_j[1]) \cdot \dots \cdot \xi(x_j[|x_j| - 1])$ for each $x_j \in X$.

Note that the length of each edge label is divisible by $q := 2m + 2$. For $t \geq 0$, let $T_t := [q \cdot t, q \cdot (t + 1) - 1]$ denote the t -th time block, that is, the q consecutive time steps starting from $q \cdot t$. Note that the j -th edge label limited to the t -th time block $\tau((v_{j-1}, v_j))[T_t]^\circ$ is exactly $\xi(x_j[t]^\circ)$. Next, we show that X is a yes-instance of PERIODIC CHARACTER ALIGNMENT if and only if \mathcal{G} is a yes-instance of PERIODIC COP & ROBBER.

(\Rightarrow) Let t be a position such that $x[t]^\circ = 1$ for all $x \in X$. We describe the winning strategy for the cop. The cop should choose the vertex v_0 as her start vertex and should never move until the beginning $t^* := q \cdot t$ of the t -th time block. By construction and the fact that $x_i[t]^\circ = 1$ for each $x_i \in X$, $\tau((v_{i-1}, v_i))[T_t]^\circ = \xi(1) = 1^m 01^m 1$. Hence, the cop can move from vertex v_i to vertex v_{i+1} in time step $t^* + i$ for each $i \in [0, m - 1]$ and, thus, reach the vertex v_m in time step $t^* + m - 1$. Moreover, the cop can then move to the vertex s in time step $t^* + m$. By construction, $\tau((s, v_0))[t^* + j]^\circ = 0$ for each $j \in [0, m]$. Hence, the cop has a winning strategy since she started at vertex v_0 and moved over every vertex of V while the robber was not able to traverse the edge (s, v_0) .

(\Leftarrow) Suppose that for every position t , there is some $x_j \in X$ with $x_j[t]^\circ = 0$. We show that the robber has a winning strategy. For some time step, let w_C and w_R denote the vertex of the cop, respectively robber in this time step. We call the vertex v_0 safe for all vertices of $V \setminus \{v_0, s\}$, we call v_m safe for v_0 and s , and we call s safe for v_0 . Let u_C be the start vertex of the cop, then the robber should choose a vertex which is safe for u_C as his start vertex.

▷ **Claim 4.** Let $t^* = t \cdot q$ be the beginning of the t -th time block for some $t \geq 0$, let u_C be the vertex of the cop at time step t^* and u_R be the vertex of the robber at time step t^* . If u_R is safe for u_C , then the robber has a strategy such that the cop cannot catch him in the t -th time block and the robber ends on a vertex that is safe for the vertex of the cop at the end of the t -th time block.

Proof.

Case 1: $u_C \in V \setminus \{s, v_0\}$ and $u_R = v_0$. The robber should wait on vertex v_0 until the beginning of time step $t^* + m$. Since the edge (s, v_0) only exists in the last time step of the t -th time block, the cop cannot catch the robber in any of these time steps. If the cop does not traverse the edge (v_m, s) in time step $t^* + m$, then the robber should stay on vertex v_0 until the beginning of the next time block. Since the edge (v_m, s) only exists in time steps t' with $t' \bmod q = m$, it follows that the cop ends on some vertex of $V \setminus \{s, v_0\}$ at the end of the t -th time block. Thus, at the beginning of the $(t + 1)$ -th time block, the vertex of the robber is safe for the vertex of the cop.

Otherwise, the cop traverses the edge (v_m, s) in time step $t^* + m$. Then, the robber should traverse the edge (v_{i-1}, v_i) in time step $t^* + m + i$ for each $i \in [1, m]$, while the cop has to wait on s . Hence, the robber reaches v_m in time step $t^* + q - 2$. In time step $t^* + q - 1$, the cop can either stay on s or move to v_0 . In both cases the robber should stay on v_m which is safe for both s and v_0 .

Case 2: $u_C = s$ and $u_R = v_m$. Since the edge (s, v_0) only exists in the last time step of the t -th time block, the cop has to stay on s until the beginning of time step $t^* + q - 1$. In time step $t^* + q - 1$, the cop can either stay on s or move to v_0 . In both cases the robber stays on v_m which is safe for both s and v_0 .

Case 3: $u_C = v_0$ and $u_R \in \{v_m, s\}$. Let $j \in [1, m]$ such that $x_j[t]^\circ = 0$, recall that by definition of τ it follows that $\tau((v_{j-1}, v_j))[T_t]^\circ = 0^m 01^m 1$. Thus, the cop cannot reach the vertex v_m in the first $m + 1$ time steps of the t -th time block. In time step $t^* + m$, the robber should stay on s if s is his current vertex or traverse the edge (v_m, s) , otherwise. Since this is the only time step in which this edge exists in the t -th time block, the cop cannot catch the robber in this time block. Until the beginning of time step $t^* + q - 1$, the robber should stay on s . If the cop ends her turn on vertex v_0 , then the robber should stay on s . Otherwise, the robber should traverse the edge (s, v_0) in time step $t^* + q - 1$. In both cases, the vertex of the robber is safe for the vertex of the cop at the beginning of the $(t + 1)$ -th time block. \triangleleft

By using Claim 4, one can show via induction, that the robber has an infinite evasive strategy and, thus, \mathcal{G} is a no-instance of PERIODIC COP & ROBBER. \blacktriangleleft

For the next section, we will stick to edge periodic cycles and consider families of cop-winning *undirected* edge periodic cycles.

4 Sharp bounds on the length required to ensure robber-winning edge periodic cycles

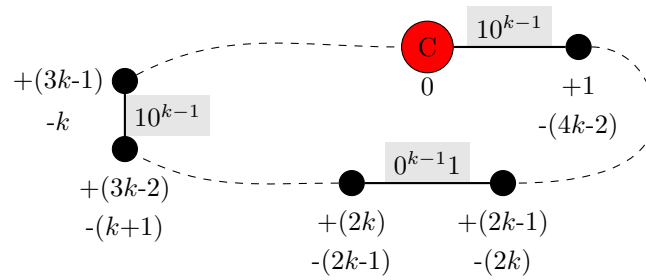
In [8], an upper bound on the cycle length of an edge periodic cycle in dependence of $\text{lcm}(L_{\mathcal{G}})$, required to ensure an robber winning strategy, was given. Namely, for $|V| = n$, the graph \mathcal{G} is robber winning if $n \geq 2 \cdot \ell \cdot \text{lcm}(L_{\mathcal{G}})$, where $\ell = 1$ if $\text{lcm}(L_{\mathcal{G}}) \geq 2 \cdot \max(L_{\mathcal{G}})$, and $\ell = 2$, otherwise ([8, Theorem 3]). So far, these bounds were not sharp, as for instance, in [8], the only lower bounds are given by cop winning strategies for families of edge periodic cycles with $n = 1.5 \cdot \text{lcm}(L_{\mathcal{G}})$ for $\ell = 1$ ([8, Theorem 5]), and $n = 3 \cdot \text{lcm}(L_{\mathcal{G}})$ for $\ell = 2$ and $\max(L_{\mathcal{G}}) = \text{lcm}(L_{\mathcal{G}})$ ([8, Theorem 4]). We show that both upper bounds ensuring a robber winning strategy are sharp by presenting infinite families of cop-winning edge periodic cycles with $n = 2 \cdot \ell \cdot \text{lcm}(L_{\mathcal{G}}) - 1$ vertices.

► **Theorem 5.** *For every $k \geq 3$ and $\ell \in \{1, 2\}$, there exists a cop-winning edge periodic cycle $\mathcal{G} = (V, E, \tau)$ with $\max(L_{\mathcal{G}}) = k$ and $n = 2 \cdot \ell \cdot \text{lcm}(L_{\mathcal{G}}) - 1$ vertices, where $\text{lcm}(L_{\mathcal{G}}) \geq 2k$ if $\ell = 1$, and $\text{lcm}(L_{\mathcal{G}}) = k$, otherwise.*

In order to prove Theorem 5, we give families of edge periodic cycles for $\ell = 1$ and $\ell = 2$, each, beginning with $\ell = 2$, i.e., the case that $\text{lcm}(L_{\mathcal{G}}) < 2 \cdot \max(L_{\mathcal{G}})$.

► **Lemma 6.** *For every $k \geq 2$ there exists an edge periodic cycle $\mathcal{G} = (V, E, \tau)$ with $\text{lcm}(L_{\mathcal{G}}) = k = \max(L_{\mathcal{G}})$, and $n = 4k - 1$ vertices that is cop-winning.*

Proof. Consider the edge periodic cycle $\mathcal{G}_k = (V, E, \tau)$ depicted in Figure 4 with $|V| = 4k - 1$. This graph admits a cop winning strategy if the cop picks the highlighted vertex with index 0 as her start vertex. The vertices are indexed by positive numbers indicating their

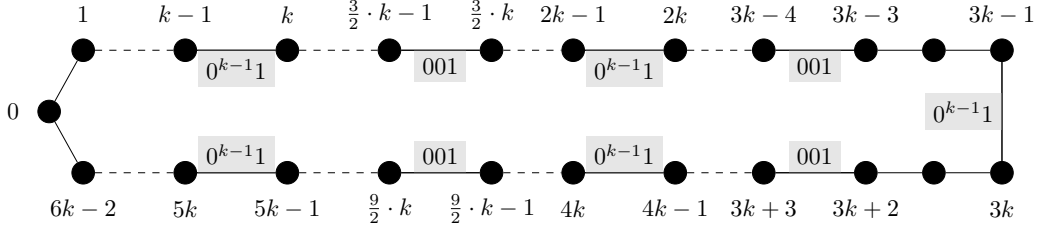


■ **Figure 4** Cycle with $4 \cdot k - 1$ vertices and $\text{lcm}(L_G) = k$ with a cop winning strategy from the start vertex marked in red. Edges not drawn (depicted by dots) are 1-edges; for all other edges, $\tau(e)$ is explicitly noted (with gray background). The clockwise [counterclockwise] distance of each vertex to the start vertex of the cop is given as a positive [negative] number.

■ **Table 1** Time steps with corresponding positions of cop and robber in the edge periodic cycle depicted in Figure 4. All positions are *after* moving in this time step. The time step s denotes the start configuration. Recall that the cop moves first.

time step	pos. cop	pos. robber	time step	pos. cop	pos. robber
s	0	$2k - 1$	s	0	$-(2k - 1)$
0	1	$2k - 1$	$k - 1$	$-(k)$	$-(2k)$
$k - 1$	k	$2k$	k	$-(k + 1)$	$-(2k + 1)$
$2k - 3$	$2k - 2$	$3k - 2$	$2k - 2$	$-(2k - 1)$	$-(3k - 1)$
$2k - 2$	$2k - 1$	$3k - 2$	$2k - 1$	$-(2k)$	$-(3k)$
$2k - 1$	$2k$	$3k - 2$	$3k - 3$	$-(3k - 2)$	$-(4k - 2)$
$2k$	$2k + 1$	$3k - 1$	$3k$	$-(3k + 1)$	0
$3k - 3$	$3k - 2$	$4k - 4$	$4k - 3$	$-(4k - 2)$	$-(k - 3)$
$3k$	$3k - 1$	0	$4k$	0	$-(k)$
$3k + 1$	$3k$	0	$5k - 1$	$-(k - 1)$	$-(k)$
$4k - 1$	$4k - 2$	0	$5k$	$-(k)$	$-(k)$
$4k$	0	$-(k)$			

clockwise distance to the start vertex of the cop, and with negative numbers indicating their counterclockwise distance. Let the cop pick vertex 0. We consider the antipolar vertices $+(2k - 1)$ and $-(2k - 1)$ as potential start vertices of the robber. We show that if the robber picks vertex $+(2k - 1)$, then the cop has a winning strategy by continuously running clockwise, starting in time step zero, and if the robber picks vertex $-(2k - 1)$, the same applies running counterclockwise. Note that these two positions represent extrema and being able to catch the robber at these vertices implies being able to catch him at all vertices in the graph. Table 1 shows the positions of the cop and robber for these strategies for $k \geq 4$. For each time step, the position after both players moved are depicted; s is the start configuration. We abbreviate consecutive 1-edges and only depict the time steps and positions when one of the players reaches a non-trivial edge. For the cases of $k = 2$ and $k = 3$ the cop catches the robber earlier than depicted in Table 1, namely in step $t = 6$ clockwise and $t = 8$ counterclockwise for $k = 2$ and in step $t = 6$ clockwise and $t = 9$ counterclockwise for $k = 3$ if the robber chooses the corresponding antipolar start vertices. Details on case $k = 2$ and $k = 3$, and a concrete example for $k = 4$, can be found in the appendix. ◀



■ **Figure 5** Cycle with $6 \cdot k - 1$ vertices and $\text{lcm}(L_{\mathcal{G}}) = 3k$ with a cop winning strategy from the start vertex 0 where $k = 2^m$ and $m \geq 2$. Edges not drawn (depicted by dots) or edges without an explicit label are 1-edges; for all other edges, $\tau(e)$ is explicitly noted (with gray background).

For the case that $\ell = 1$, i.e., when $\text{lcm}(L_{\mathcal{G}}) \geq 2 \cdot \max(L_{\mathcal{G}})$, we slightly adapt the family of graphs depicted in Figure 4. Note that for $\max(L_{\mathcal{G}}) = 2$ there is no edge periodic cycle $\mathcal{G} = (V, E, \tau)$ with $\text{lcm}(L_{\mathcal{G}}) > \max(L_{\mathcal{G}}) = 2$.

► **Lemma 7.** *For every $k \geq 3$ with $k \neq 2^m$ for all $m \in \mathbb{N}$, there exists an edge periodic cycle $\mathcal{G} = (V, E, \tau)$ with $\text{lcm}(L_{\mathcal{G}}) = 2 \cdot \max(L_{\mathcal{G}}) = 2 \cdot k$, and $n = 2 \cdot 2k - 1$ vertices that is cop-winning.*

Proof. For the case $\ell = 1$ we introduce an artificial edge label in the edge periodic cycle in Figure 4, such that the $\text{lcm}(L_{\mathcal{G}})$ is exactly $2k$. This edge will not affect the run of the cop. Its purpose is to introduce a factor 2 in the number of vertices compensating the missing factor 2 from the variable ℓ . Therefore, note that the edge $e_{1,2}$ connecting vertex $+1$ and $+2$ is taken by the cop only once, in the clockwise run in time step 1 and in the counterclockwise run in time step $4k - 3$. Hence, the cop only crosses the edge in an odd time step. We can write k as $k = 2^i \cdot j$ where j is an odd number with $j > 1$ since $k \neq 2^m$. Then, introducing a string $\tau(e_{1,2}) = 01^{2^{i+1}-1}$ of length 2^{i+1} yields a least common multiple of $\text{lcm}(L_{\mathcal{G}}) = 2^{i+1} \cdot j = 2 \cdot k$. ◀

In the case of $\max(L_{\mathcal{G}}) = k = 2^m$ for some $m \in \mathbb{N}$, it holds that for the smallest possible value of $\text{lcm}(L_{\mathcal{G}})$ with $\text{lcm}(L_{\mathcal{G}}) > \max(L_{\mathcal{G}})$, we have $\text{lcm}(L_{\mathcal{G}}) \geq 3 \cdot \max(L_{\mathcal{G}})$. Hence, in these cases we need a separate family of graphs.

► **Lemma 8.** *For every $k = 2^m$ with $m \geq 2$, there exists an edge periodic cycle $\mathcal{G} = (V, E, \tau)$ with $\text{lcm}(L_{\mathcal{G}}) = 3 \cdot \max(L_{\mathcal{G}}) = 3 \cdot k$, and $n = 6 \cdot k - 1$ vertices that is cop-winning.*

Proof. Consider the edge periodic cycle $\mathcal{G}_k = (V, E, \tau)$ depicted in Figure 5 with $|V| = 6k - 1$. This graph admits a cop winning strategy if the cop picks the highlighted vertex with index 0 as her start vertex. The vertices are indexed by positive numbers indicating their clockwise distance to the start vertex of the cop. Let the cop pick vertex 0. We show that if the robber picks vertex $3k - 1$, then the cop has a winning strategy by continuously running clockwise, starting in time step zero. Since for each j , starting from vertex 0, the label of the j -th edge clockwise is equal to the label of the j -th edge counterclockwise, the same applies running counterclockwise if the robber picks vertex $3k$. Note that these two positions represent extrema and being able to catch the robber at these vertices implies being able to catch him at all vertices in the graph. Suppose that the robber picks vertex $3k - 1$. Since $k = 2^m$ for some $m \geq 2$, $\frac{3}{2} \cdot k$ and $\frac{9}{2} \cdot k$ are divisible by 3. Hence for each $j \in [1, 6k - 3]$, the cop can traverse the edge $\{j, j + 1\}$ in time step j and, thus, reach the vertex $5k - 1$ in time step $5k - 2$. We show that, starting from vertex $3k - 1$ and running clockwise, the robber cannot reach vertex $5k$ prior than time step $5k - 1$. This then implies, that the cop catches

the robber after at most $5k - 2$ time steps. Note that the first time the robber can traverse the edge $\{3k - 1, 3k\}$ is at time step $k - 1$. Hence, the robber reaches the vertex $3k + 2$ not prior than time step $k + 1$. Since k is not divisible by 3, the robber cannot traverse the edge $\{3k + 2, 3k + 3\}$ in time step $k + 2$. Thus, the robber cannot reach the vertex $4k - 1$ prior than time step $2k$ and consequently, he cannot traverse the edge $\{4k - 1, 4k\}$ prior than time step $3k - 1$. Hence, the robber reaches the vertex $\frac{9}{2}k - 1$ not prior than time step $\frac{7}{2}k - 2$. Since k is not divisible by 3, the robber cannot traverse the edge $\{\frac{9}{2}k - 1, \frac{9}{2}k\}$ in time step $\frac{7}{2}k - 1$. Thus, the robber cannot reach the vertex $5k - 1$ prior than time step $4k$ and consequently, he cannot traverse the edge $\{5k - 1, 5k\}$ prior than time step $5k - 1$. Hence, the statement holds. A concrete example for $k = 4$ can be found in the appendix. ◀

5 Complexity upper bounds

The main result of this section is that the PERIODIC COP & ROBBER problem for *general* edge periodic graphs can be solved in polynomial space. Note that two-player games may take exponentially many turns and hence containment in PSPACE is not obvious. In our case, already the period of graphs on which the game is played is exponential in general. This prohibits a standard incremental PSPACE algorithm approach. We show that despite the potentially exponential period of the sequence of graphs $\mathcal{G}(t)$ we show that we can determine whether the cop has a winning strategy by sweeping through the configuration space in a way that we only consider polynomially many vertices in each step. The fact that we only consider one cop and one robber is here crucial for the polynomial bound.

► **Theorem 9.** *PERIODIC COP & ROBBER for edge periodic graphs is contained in PSPACE.*

For general edge periodic graphs, the PERIODIC COP & ROBBER problem was reduced in [8] to a variant of the AND-OR GRAPH REACHABILITY problem [6] via an exponential time reduction. The AND-OR GRAPH REACHABILITY problem is a two player game where players move a token in an AND-OR graph from a source to a target. An AND-OR graph is a graph $G = (V_\wedge \cup V_\vee, E)$ where the set of vertices is partitioned into a set of AND vertices V_\wedge and a set of OR vertices V_\vee . If the token is on an OR vertex, then player 0 moves the token, otherwise player 1 moves the token. Player 0 wins if the token finally reaches the target. The AND-OR GRAPH REACHABILITY problem is known to be PTIME-complete [12]. The reduction in [8] unrolls the edge periodic graph into its configuration graph. Considering the PERIODIC COP & ROBBER problem, we define for an edge periodic graph $\mathcal{G} = (V, E, \tau)$, the *configuration graph* $\mathcal{C}_\mathcal{G} = (V_{\mathcal{C}_\mathcal{G}}, E_{\mathcal{C}_\mathcal{G}})$ of \mathcal{G} with node set $V \times V \times \{c, r\} \times [\text{lcm}(L_\mathcal{G})]$. Here, a node (u_c, u_r, s, t) denotes in the temporal snapshot graph $\mathcal{G}(t)$, that the cop is on vertex u_c , the robber on vertex u_r , and the cop moves next if $s = c$, otherwise the robber moves next. A node (u_c, u_r, s, t) is connected with a directed edge to some node $(u'_c, u'_r, s', (t + 1) \bmod \text{lcm}(L_\mathcal{G}))$, if $s = c$, $s' = r$, $u'_r = u_r$, and $u'_c = u_c$ or u_c and u'_c are connected by an edge which is present in time step t . If $s = r$ the same holds for c and r interchanged. Hence, the configuration graph $\mathcal{C}_\mathcal{G}$ is of size $\mathcal{O}(|V|^2 \cdot \text{lcm}(L_\mathcal{G}))$.

The configuration graph $\mathcal{C}_\mathcal{G}$ is then turned into an AND-OR graph by declaring nodes (u_c, u_r, s, t) with $s = c$ as OR vertices and nodes with $s = r$ as AND vertices. As $\text{lcm}(L_\mathcal{G})$ can be of size exponential in $|L_\mathcal{G}|$, it follows from [8] that PERIODIC COP & ROBBER is contained in EXPTIME. Note that the exponential blow-up comes from the very deterministic process of unrolling the edge periodic graph into a TVG with global periodicity $\text{lcm}(L_\mathcal{G})$. Representing this TVG in a framework which only allows for global periodicity, such as the setting in [18], would require a representation of the TVG in size $\mathcal{O}(\text{lcm}(L_\mathcal{G}))$ and allow for a polynomial time reduction to the AND-OR GRAPH REACHABILITY problem yielding membership in PTIME for PERIODIC COP & ROBBER in this setting.

In the following, we show how use the structural properties of the configuration graph $\mathcal{C}_{\mathcal{G}}$ to solve the AND-OR GRAPH REACHABILITY for $\mathcal{C}_{\mathcal{G}}$ in polynomial space. We begin with giving an upper bound on the length of a shortest chase in \mathcal{G} .

► **Lemma 10.** *Let $\mathcal{G} = (V, E, \tau)$ be an edge periodic graph. If \mathcal{G} is cop-winning, then the robber can be caught within $n^2 \cdot \text{lcm}(L_{\mathcal{G}})$ rounds.*

Proof. Consider the configuration graph $\mathcal{C}_{\mathcal{G}} = (V_{\mathcal{C}_{\mathcal{G}}}, E_{\mathcal{C}_{\mathcal{G}}})$ of \mathcal{G} . Note that two configurations of a PERIODIC COP & ROBBER game on the same temporal snapshot graph with identical positions of the cop and robber in different time steps t and t' which differ by a multiple of $\text{lcm}(L_{\mathcal{G}})$ are indistinguishable as configurations of the game. Hence, the size of $\mathcal{C}_{\mathcal{G}}$ is bounded by $2n^2 \cdot \text{lcm}(L_{\mathcal{G}})$.

Now let π be a shortest sequence of configurations for the start vertices v_c and v_r such that π ends with a cop-winning configuration. If $|\pi| > 2n^2 \cdot \text{lcm}(L_{\mathcal{G}})$, then π contains two indistinguishable configurations π_{i_1} and π_{i_2} . Clearly, the cop made no progress towards capturing the robber in the sequence $\pi_{i_1}, \dots, \pi_{i_2-1}$. As π_{i_1} and π_{i_2} are indistinguishable, removing the sequence $\pi_{i_1}, \dots, \pi_{i_2-1}$ from π yields a shorter sequence of configurations ending in a cop-winning configurations violating the assumed minimality of π . ◀

Proof of Theorem 9. We will follow the ideas from [8] of reducing the PERIODIC COP & ROBBER problem to a variant of the AND-OR GRAPH REACHABILITY problem. Therefore, we will need the notion of *attractors* in an AND-OR graph. Let $G = (V, E)$ be an AND-OR graph with $V = V_{\wedge} \cup V_{\vee}$. Instead of considering a single target, we consider a set T of targets and say that player 0 wins if the token finally reaches any state in T . Let $s \in V$ be the start vertex. Intuitively, the set of attractors of G is the set of vertices from which player 0 can win the game. More formally, we inductively define the set of attractors Attr of T as:

$$\begin{aligned} \text{Attr}^0 &= T, \\ \text{Attr}^{i+1} &= \text{Attr}_i \cup \{v \in V_{\wedge} \mid \forall \{u, v\} \in E: u \in \text{Attr}^i\} \cup \\ &\quad \{v \in V_{\vee} \mid \exists \{u, v\} \in E: u \in \text{Attr}^i\}, \\ \text{Attr} &= \bigcup_{i \geq 0} \text{Attr}^i. \end{aligned}$$

Let $\mathcal{G} = (V, E, \tau)$ be the input edge periodic graph and let $\mathcal{C}_{\mathcal{G}} = (V_{\mathcal{C}_{\mathcal{G}}}, E_{\mathcal{C}_{\mathcal{G}}})$ be the configuration graph of \mathcal{G} . We will also identify $\mathcal{C}_{\mathcal{G}}$ as an AND-OR graph by declaring nodes (u_c, u_r, s, t) of $\mathcal{C}_{\mathcal{G}}$ with $s = c$ as OR vertices and nodes with $s = r$ as AND vertices. We define the set of nodes $T = \{(u_c, u_r, s, t) \in V_{\mathcal{C}_{\mathcal{G}}} \mid u_c = u_r\}$ as the target set of the AND-OR graph $\mathcal{C}_{\mathcal{G}}$. We will now prove that the ideas from [8] of solving the PERIODIC COP & ROBBER game by checking if (i) there is some vertex u_c such that for all vertices $u_r \in V$, the node $(u_c, u_r, c, 0) \in V_{\mathcal{C}_{\mathcal{G}}}$ is an attractor in the AND-OR graph $\mathcal{C}_{\mathcal{G}}$, can be implemented in polynomial space. Note that the set of attractors in $\mathcal{C}_{\mathcal{G}}$ corresponds to the set of configurations from which the cop has a winning strategy. We use Lemma 10 to unroll the configuration graph in order to obtain a leveled directed acyclic graph (DAG) which has width n^2 in each level and through which we can sweep level by level in order to verify property (i).

By Lemma 10 we know that in order to verify property (i) it is sufficient to consider paths of length at most $2n^2 \cdot \text{lcm}(L_{\mathcal{G}})$ in $\mathcal{C}_{\mathcal{G}}$ (the factor 2 is due to the alternation of players). As the configuration graph is cyclic (due to modulo counting by $\text{lcm}(L_{\mathcal{G}})$) we unroll the graph n^2 times to allow for different time steps up to $n^2 \cdot \text{lcm}(L_{\mathcal{G}})$. The obtained DAG is big enough to contain any shortest chase starting in any time step $t \leq \text{lcm}(L_{\mathcal{G}})$. The so obtained graph $\mathcal{C}'_{\mathcal{G}}$ consists of the node set $V'_{\mathcal{C}_{\mathcal{G}}} = V \times V \times \{c, r\} \times [n^2 \cdot \text{lcm}(L_{\mathcal{G}})]$ and the edge set $E'_{\mathcal{C}_{\mathcal{G}}}$

extending $E_{\mathcal{G}}$ as $((u_c, u_r, s, t), (u'_c, u'_r, s', t')) \in E_{\mathcal{C}'_{\mathcal{G}}}$ if and only if $t' \in \{t, t+1\}$, $((u_c, u_r, s, t \bmod \text{lcm}(L_{\mathcal{G}})), (u'_c, u'_r, s', t' \bmod \text{lcm}(L_{\mathcal{G}}))) \in E_{\mathcal{C}'_{\mathcal{G}}}$ and $t, t' \leq n^2 \cdot \text{lcm}(L_{\mathcal{G}}) - 1$. Verifying property (i) then corresponds to a reachability game in the AND-OR graph associated with $\mathcal{C}'_{\mathcal{G}}$ with target set $T = \{(u_c, u_r, s, t) \in V_{\mathcal{C}'_{\mathcal{G}}} \mid u_c = u_r\}$ which can be solved using the notion of attractors. Note that in $\mathcal{C}'_{\mathcal{G}}$ only nodes with identical time steps or t and $t+1$ are connected. Hence, in order to compute which nodes with time step t belong to the set of attractors, we need to only know which nodes with time step $t+1$ are attractors. Since $\mathcal{C}'_{\mathcal{G}}$ is a DAG we can start in the level with $t = n^2 \cdot \text{lcm}(L_{\mathcal{G}}) - 1$ of $\mathcal{C}'_{\mathcal{G}}$.

$$\begin{aligned} \text{Attr}_r^{n^2 \cdot \text{lcm}(L_{\mathcal{G}}) - 1} &:= \{(u_c, u_r, r, n^2 \cdot \text{lcm}(L_{\mathcal{G}}) - 1) \mid u_c = u_r\}, \\ \text{Attr}_r^t &:= \{(u_c, u_r, r, t) \mid \forall ((u_c, u_r, r, t), (u'_c, u'_r, c, t+1)) \in E_{\mathcal{C}'_{\mathcal{G}}} : (u'_c, u'_r, c, t+1) \\ &\quad \in \text{Attr}_c^{t+1}\} \cup \{(u_c, u_r, r, t) \mid u_c = u_r\}, \text{ for } n^2 \cdot \text{lcm}(L_{\mathcal{G}}) - 2 \geq t \geq 0, \\ \text{Attr}_c^t &:= \{(u_c, u_r, c, t) \mid \exists ((u_c, u_r, c, t), (u'_c, u'_r, r, t)) \in E_{\mathcal{C}'_{\mathcal{G}}} : (u'_c, u'_r, r, t) \in \text{Attr}_r^t\} \\ &\quad \cup \{(u_c, u_r, c, t) \mid u_c = u_r\}, \text{ for } n^2 \cdot \text{lcm}(L_{\mathcal{G}}) - 1 \geq t \geq 0. \end{aligned}$$

For each level $n^2 \cdot \text{lcm}(L_{\mathcal{G}}) - 1 \geq t \geq 0$ we only need to keep the last level ($t+1$ if existent)¹ of $\mathcal{C}'_{\mathcal{G}}$ in memory in order to compute the sets Attr_c^t and Attr_r^t of nodes (u_c, u_r, s, t) in $\mathcal{C}'_{\mathcal{G}}$ from which the cop has a winning strategy where s equals the sub-script. Note that $\bigcup_{0 \leq t \leq n^2 \cdot \text{lcm}(L_{\mathcal{G}}) - 1} \text{Attr}_c^t \cup \text{Attr}_r^t = \text{Attr}$. In order to verify property (i) we only need to keep the current and latest sets $\text{Attr}_c^t, \text{Attr}_r^t, \text{Attr}_c^{t+1}, \text{Attr}_r^{t+1}$ in memory yielding a polynomial space algorithm. ◀

6 Discussion

While we improved the currently known upper bound for the PERIODIC COP & ROBBER problem on edge periodic graphs from EXPTIME to PSPACE and improved the lower bounds, of being NP-hard, to include also the very restrictive classes of directed and undirected edge periodic cycles, a gap in the complexity of PERIODIC COP & ROBBER remains. It is worth noticing that on one side the chosen representation of edge periodic graphs is quite compact, as a natural proof for a cop-winning strategy might be of exponential length in the input size, since the periodicity of the whole graph is the least common multiple of the periodicity of each edge, which prevents the use of a simple guess & check approach for NP-membership. On the other side, the chosen representation is still exponentially larger than the representation by on-line programs used in [17] where PSPACE-completeness for the reachability problem on a related but different class of periodic TVGs was obtained.

If we consider *directed* edge periodic *cycles*, then determining whether the given cycle is cop-winning boils down to deterministically simulating the chase starting from a (guessed) cop vertex and time step, as the optimal strategies for the cop and robber are both to keep running whenever possible (without bumping into the cop). For the robber the optimal start vertex is directly behind the cop. Since $\text{lcm}(L_{\mathcal{G}})$ can be exponentially large in the size of \mathcal{G} the only known upper bound on the number of steps in the simulation of the chase starting in time step t is exponential in the size of \mathcal{G} , while the chase does not reveal any complexity. The simulation could even be performed by a log-space Turing-Machine being equipped with

¹ Note that we can easily compute the snapshot $\mathcal{G}(n^2 \cdot \text{lcm}(L_{\mathcal{G}})) = \mathcal{G}(0)$ by drawing all edges with $\tau(e)[0]^{\circ} = 1$; and from $\mathcal{G}(t)$ for some time step t , the snapshot $\mathcal{G}(t-1)$ by shifting the pointer in each $\tau(e)$ one step to the left. Therefore, we can compute from each level $t+1$ of $\mathcal{C}'_{\mathcal{G}}$ the level t in polynomial time and space.

a clock which allows for modulo queries of logarithmic size. To better understand the precise complexity of PERIODIC COP & ROBBER on directed edge periodic cycles, the theoretical analysis of potential families of cycles with shortest cop-winning strategies of exponential size would be of great interest and might indicate the necessity for a new complexity class consisting of simple simulation problems with exponential duration time.

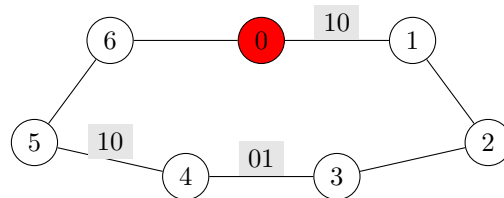
References

- 1 Sandeep Bhadra and Afonso Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In *Proceedings of the 2nd International Conference on Ad-Hoc, Mobile, and Wireless Networks*, volume 2865 of *Lecture Notes in Computer Science*, pages 259–270. Springer, 2003.
- 2 Niclas Boehmer, Vincent Froese, Julia Henkel, Yvonne Lasars, Rolf Niedermeier, and Malte Renken. Two influence maximization games on graphs made temporal. *CoRR*, abs/2105.05987, 2021. [arXiv:2105.05987](https://arxiv.org/abs/2105.05987).
- 3 Anthony Bonato. *The game of cops and robbers on graphs*. American Mathematical Soc., 2011.
- 4 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *Int. J. Parallel Emergent Distributed Syst.*, 27(5):387–408, 2012.
- 5 Nancy E. Clarke and Gary MacGillivray. Characterizations of k-copwin graphs. *Discret. Math.*, 312(8):1421–1425, 2012.
- 6 Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. *Theor. Comput. Sci.*, 386(3):188–217, 2007.
- 7 Bolin Ding, Jeffrey Xu Yu, and Lu Qin. Finding time-dependent shortest paths over large graphs. In *Proceedings of the 11th International Conference on Extending Database Technology*, volume 261 of *ACM International Conference Proceeding Series*, pages 205–216. ACM, 2008.
- 8 Thomas Erlebach and Jakob T. Spooner. A game of cops and robbers on graphs with periodic edge-connectivity. In *Proceedings of 46th International Conference on Current Trends in Theory and Practice of Informatics*, volume 12011 of *Lecture Notes in Computer Science*, pages 64–75. Springer, 2020.
- 9 Niloy Ganguly, Andreas Deutsch, and Animesh Mukherjee. Dynamics on and of complex networks. *Applications to Biology, Computer Science, and the Social Sciences*, 2009.
- 10 Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):1–30, 2015.
- 11 Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.
- 12 Neil Immerman. Number of quantifiers is better than number of tape cells. *J. Comput. Syst. Sci.*, 22(3):384–406, 1981.
- 13 Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. *Theor. Comput. Sci.*, 634:1–23, 2016.
- 14 Nils Morawietz, Carolin Rehs, and Mathias Weller. A timecop's work is harder than you think. In *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science*, volume 170 of *LIPICs*, pages 71:1–71:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 15 Richard J. Nowakowski and Peter Winkler. Vertex-to-vertex pursuit in a graph. *Discret. Math.*, 43(2-3):235–239, 1983.
- 16 Alain Quilliot. *Jeux et pointes fixes sur les graphes*. PhD thesis, Ph. D. Dissertation, Université de Paris VI, 1978.
- 17 Klaus Sutner and Wolfgang Maass. Motion planning among time dependent obstacles. *Acta Informatica*, 26(1-2):93–122, 1988.

- 18 Klaus Wehmuth, Artur Ziviani, and Eric Fleury. A unifying model for representing time-varying graphs. In *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015*, pages 1–10. IEEE, 2015.
- 19 Zhensheng Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Commun. Surv. Tutorials*, 8(1-4):24–37, 2006.

A Details on Lemma 6

We explicitly give the edge periodic cycles for $k = 2$, $k = 3$, and $k = 4$ in the proof of Lemma 6. For $k = 2$ and $k = 3$ the chase of the cop will be shorter than described in Table 1 and for $k \geq 4$ the chase will be exactly as described in general in Table 1. The edge periodic cycle for $k = 2$ is depicted in Figure 6 and the chase is described in Table 2. For $k = 3$ the edge periodic cycle is depicted in Figure 7 and the chase is described in Table 3. Finally, for $k = 4$, the edge periodic cycle is depicted in Figure 8 and the explicit chase is described in Table 4. Note that Table 4 is identical to Table 1 if we set $k = 4$ in Table 1.

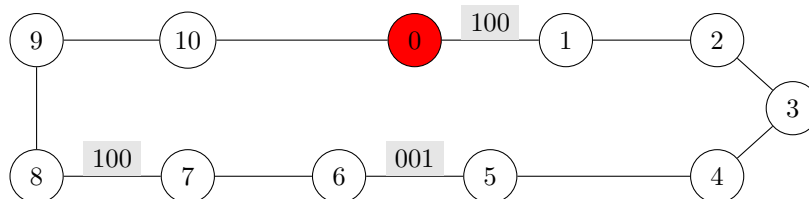


■ **Figure 6** Edge periodic cycle for the case $k = 2$ in Lemma 6 with $4 \cdot k - 1 = 7$ vertices and $\text{lcm}(L_{\mathcal{G}}) = 2$ with a cop winning strategy from the start vertex marked in red. Edges without edge label are 1-edges; for all other edges, $\tau(e)$ is explicitly noted (with gray background).

■ **Table 2** Time steps with corresponding positions of cop and robber in the edge periodic cycle depicted in Figure 6. All positions are *after* moving in this time step. The time step s denotes the start configuration. Recall that the cop moves first.

time step	pos. cop	pos. robber	time step	pos. cop	pos. robber
s	0	3	s	0	4
0	1	3	0	6	4
1	2	4	1	5	3
2	3	5	2	4	2
3	4	6	3	3	1
4	5	0	4	2	0
5	6	0	5	1	6
6	0	0	6	0	5
		0	7	6	5
		0	8	5	0

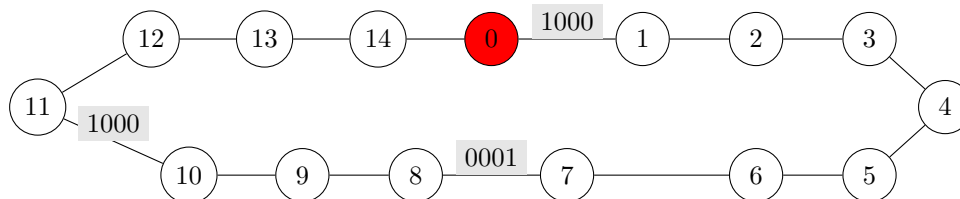
77:16 A Timecop's Chase Around the Table



■ **Figure 7** Edge periodic cycle for the case $k = 3$ in Lemma 6 with $4 \cdot k - 1 = 11$ vertices and $\text{lcm}(L_G) = 3$ with a cop winning strategy from the start vertex marked in red. Edges without edge label are 1-edges; for all other edges, $\tau(e)$ is explicitly noted (with gray background).

■ **Table 3** Time steps with corresponding positions of cop and robber in the edge periodic cycle depicted in Figure 7. All positions are *after* moving in this time step. The time step s denotes the start configuration. Recall that the cop moves first.

time step	pos. cop	pos. robber	time step	pos. cop	pos. robber
s	0	5	s	0	6
0	1	5	0	10	6
1	2	5	1	9	6
2	3	6	2	8	5
3	4	7	3	7	4
4	5	7	4	6	3
5	6	7	5	5	2
6	7	8	6	4	1
			7	3	1
			8	2	1
			9	1	0



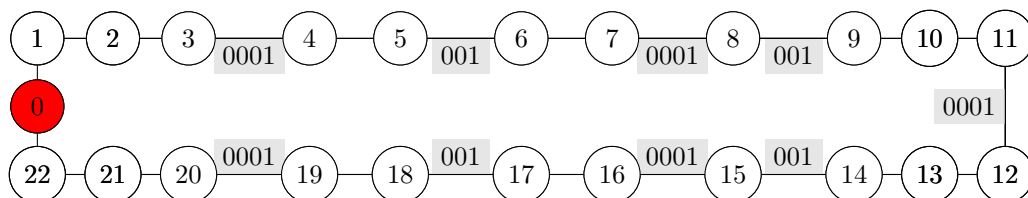
■ **Figure 8** Edge periodic cycle for the case $k = 4$ in Lemma 6 with $4 \cdot k - 1 = 15$ vertices and $\text{lcm}(L_G) = 4$ with a cop winning strategy from the start vertex marked in red. Edges without edge label are constant 1-edges; for all other edges, $\tau(e)$ is explicitly noted (with gray background).

■ **Table 4** Time steps with corresponding positions of cop and robber in the edge periodic cycle depicted in Figure 8. Note that the position of the cop and robber are as described in Table 1 for the general case of $k \geq 4$. All positions are *after* moving in this time step. The time step s denotes the start configuration. Recall that the cop moves first.

time step	pos. cop	pos. robber	time step	pos. cop	pos. robber
s	0	7	s	0	8
0	1	7	0	14	8
1	2	7	1	13	8
2	3	7	2	12	8
3	4	8	3	11	7
4	5	9	4	10	6
5	6	10	5	9	5
6	7	10	6	8	4
7	8	10	7	7	3
8	9	11	8	6	2
9	10	12	9	5	1
10	10	13	10	4	1
11	10	14	11	3	1
12	11	0	12	2	0
13	12	0	13	1	14
14	13	0	14	1	13
15	14	0	15	1	12
16	0	0	16	0	11
			17	14	11
			18	13	11
			19	12	11
			20	11	0

B Details on Lemma 8

We explicitly give the edge periodic cycle for $k = 4$ in the proof of Lemma 8. The edge periodic cycle is depicted in Figure 9 and the explicit chase is described in Table 5.



■ **Figure 9** Cycle with $23 = 6k - 1$ vertices and $\text{lcm}(L_G) = 12 = 3k$ with a cop winning strategy from the start vertex 0 where $k = 4$. Edges without an explicit label are 1-edges.

77:18 A Timecop's Chase Around the Table

■ **Table 5** Time steps with corresponding positions of cop and robber in the edge periodic cycle depicted in Figure 9. All positions are *after* moving in this time step. The time step s denotes the start configuration. Recall that the cop moves first.

time step	pos. cop	pos. robber	time step	pos. cop	pos. robber
s	0	11	s	0	12
0	1	11	0	22	12
1	2	11	1	21	12
2	3	11	2	20	12
3	4	12	3	19	11
4	5	13	4	18	10
5	6	14	5	17	9
6	7	14	6	16	9
7	8	14	7	15	9
8	9	15	8	14	8
9	10	15	9	13	8
10	11	15	10	12	8
11	12	16	11	11	7
12	13	17	12	10	6
13	14	17	13	9	6
14	15	18	14	8	5
15	16	19	15	7	4
16	17	19	16	6	4
17	18	19	17	5	4
18	19	Ⓝ	18	4	Ⓝ

Syntactic Minimization Of Nondeterministic Finite Automata

Robert S. R. Myers ✉

London, United Kingdom

Henning Urbat ✉ 

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Nondeterministic automata may be viewed as succinct programs implementing deterministic automata, i.e. complete specifications. Converting a given deterministic automaton into a small nondeterministic one is known to be computationally very hard; in fact, the ensuing decision problem is PSPACE-complete. This paper stands in stark contrast to the status quo. We restrict attention to *subatomic* nondeterministic automata, whose individual states accept unions of syntactic congruence classes. They are general enough to cover almost all structural results concerning nondeterministic state-minimality. We prove that converting a monoid recognizing a regular language into a small subatomic acceptor corresponds to an NP-complete problem. The NP certificates are solutions of simple equations involving relations over the syntactic monoid. We also consider the subclass of *atomic* nondeterministic automata introduced by Brzozowski and Tamm. Given a deterministic automaton and another one for the reversed language, computing small atomic acceptors is shown to be NP-complete with analogous certificates. Our complexity results emerge from an algebraic characterization of (sub)atomic acceptors in terms of deterministic automata with semilattice structure, combined with an equivalence of categories leading to succinct representations.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Algebraic language theory, Nondeterministic automata, NP-completeness

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.78

Related Version *Full Version*: <http://arxiv.org/abs/2107.03229>

Funding *Henning Urbat*: Supported by Deutsche Forschungsgemeinschaft (DFG) under project SCHR 1118/15-1.

1 Introduction

Regular languages arise from a multitude of different perspectives: operationally via finite-state machines, model-theoretically via monadic second-order logic, and algebraically via finite monoids. In practice, *deterministic* finite automata (dfas) and *nondeterministic* finite automata (nfas) are two of the most common representations. Although the former may be exponentially larger than the latter, there is no known efficient procedure for converting dfas into small nfas, e.g. state-minimal ones. Jiang and Ravikumar proved the corresponding decision problem (does an equivalent nfa with a given number of states exist?) to be PSPACE-complete [14, 15], suggesting that exhaustively enumerating candidates is necessary. One possible strategy towards tractability is to restrict the target automata to suitable *subclasses* of nfas. The challenge is to identify subclasses permitting more efficient computation (e.g. lowering the PSPACE bound to an NP bound, enabling the use of SAT solvers), while still being general enough to cover succinct acceptors of regular languages.

In our present paper we will show that the class of *subatomic nfas* naturally meets the above requirements. An nfa accepting the language L is subatomic if each individual state accepts a union of syntactic congruence classes of L . In recent work [26] we observed that almost all known results on the structure of small nfas, e.g. for unary [6, 13], bideterministic [30],



© Robert S. R. Myers and Henning Urbat;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 78; pp. 78:1–78:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

topological [1] and biRFSA languages [19], implicitly construct small subatomic nfes. This firmly indicates that the latter form a rich class of acceptors despite their seemingly restrictive definition, i.e. in many settings computing small nfes amounts to computing small subatomic ones. Restricting to subatomic nfes yields useful additional structure; in fact, their theory is tightly linked to the algebraic theory of regular languages and the representation theory of monoids. This suggests an *algebraic* counterpart of the dfa to nfa conversion problem: given a finite monoid recognizing some regular language, compute an equivalent small subatomic nfa. Denoting its decision version (does an equivalent subatomic nfa with a given number of states exist?) by $\mathbf{MON} \rightarrow \mathbf{NFA}_{\text{syn}}$, our main result is:

► **Theorem.** The problem $\mathbf{MON} \rightarrow \mathbf{NFA}_{\text{syn}}$ is NP-complete.

In addition we also investigate *atomic nfes*, a subclass of subatomic nfes earlier introduced by Brzozowski and Tamm [4]. Similar to the subatomic case, their specific structure naturally invokes the problem of converting a pair of dfas accepting mutually reversed languages into a small atomic nfa. Denoting its decision version by $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}_{\text{atm}}$, we get:

► **Theorem.** The problem $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}_{\text{atm}}$ is NP-complete.

The short certificates witnessing that both problems are in NP are solutions of *equations* involving relations over the syntactic congruence or the Nerode left congruence, respectively.

The above two theorems sharply contrast the PSPACE-completeness of the general dfa to nfa conversion problem, but also previous results on its sub-PSPACE variants. The latter are either concerned with particular regular languages such as finite or unary ones [11, 13], or with target nfes admitting only very weak forms of nondeterminism, such as unambiguous automata [15] or dfas with multiple initial states [22]. In contrast, our present work applies to all regular languages and the restriction to (sub)atomic nfes is a purely *semantic* one.

Our results are fundamentally based upon a category-theoretic perspective on atomic and subatomic acceptors. At its heart are two equivalences of categories as indicated below:

$$\mathbf{JSL}_f^{\text{op}} \xleftarrow[\text{Structure theory}]{\simeq} \mathbf{JSL}_f \xleftarrow[\text{Complexity theory}]{\simeq} \mathbf{Dep}.$$

As shown in [26], the *structure theory* of (sub)atomic nfes emerges by interpreting them as dfes endowed with semilattice structure, and relating them to their dual automata under the familiar self-duality of the category \mathbf{JSL}_f of finite semilattices. Similarly, the *complexity theory* of (sub)atomic nfes developed in the present paper rests on the equivalence between \mathbf{JSL}_f and a category \mathbf{Dep} (see Definition 3.1) that yields succinct relational representations of finite semilattices by their irreducible elements. To derive the NP-completeness theorems, we reinterpret semilattice automata associated to (sub)atomic nfes inside \mathbf{Dep} . We regard this conceptually simple and natural categorical approach as a key contribution of our paper.

2 Atomic and Subatomic NFAs

We start by setting up the notation and terminology used in the rest of the paper, including the key concept of a (sub)atomic nfa that underlies our complexity results. Readers are assumed to be familiar with basic category [21].

Semilattices. A (*join-*)*semilattice* is a poset (S, \leq_S) in which every finite subset $X \subseteq S$ has a least upper bound (a.k.a. join) $\bigvee X$. A *morphism* between semilattices is a map preserving finite joins. If S is finite as we often assume, every subset $X \subseteq S$ also has a greatest lower

bound (a.k.a. meet) $\bigwedge X$, given by the join of its lower bounds. In particular, S has a least element $\perp_S = \bigvee \emptyset$ and a greatest element $\top_S = \bigwedge \emptyset$. An element $j \in S$ is *join-irreducible* if $j = \bigvee X$ implies $j \in X$ for every subset $X \subseteq S$. Dually, $m \in S$ is *meet-irreducible* if $m = \bigwedge X$ implies $m \in X$. We put

$$J(S) = \{ j \in S : j \text{ is join-irreducible} \} \quad \text{and} \quad M(S) = \{ m \in S : m \text{ is meet-irreducible} \}.$$

Note $\perp_S \notin J(S)$ and $\top_S \notin M(S)$. The join-irreducibles form the least set of *join-generators* of S , i.e. every element of S is a join of elements from $J(S)$, and every other subset $J \subseteq S$ with that property contains $J(S)$. Dually, $M(S)$ is the least set of *meet-generators* of S .

Let $\mathbb{2} = \{0, 1\}$ be the two-element semilattice with $0 \leq 1$. Morphisms $i: \mathbb{2} \rightarrow S$ correspond to elements of S via $i \mapsto i(1)$. Morphisms $f: S \rightarrow \mathbb{2}$ correspond to *prime filters* via $f \mapsto f^{-1}[1]$. If S is finite, these are precisely the subsets $F_{s_0} = \{s \in S : s \not\leq_S s_0\}$ for any $s_0 \in S$.

We denote by **JSL** the category of join-semilattices and their morphisms. Its full subcategory **JSL_f** of finite semilattices is self-dual [17]: there is an equivalence functor

$$\mathbf{JSL}_f^{\text{op}} \xrightarrow{\cong} \mathbf{JSL}_f$$

mapping (S, \leq_S) to the *opposite semilattice* $S^{\text{op}} = (S, \geq_S)$ obtained by reversing the order, and a morphism $f: S \rightarrow T$ to the morphism $f_*: T^{\text{op}} \rightarrow S^{\text{op}}$ sending $t \in T$ to the \leq_S -greatest element $s \in S$ with $f(s) \leq_T t$. Thus, f and f_* satisfy the adjoint relationship

$$f(s) \leq_T t \quad \text{iff} \quad s \leq_S f_*(t)$$

for all $s \in S$ and $t \in T$. The morphism f is injective (equivalently a **JSL_f**-monomorphism) iff f_* is surjective (equivalently a **JSL_f**-epimorphism).

Relations. A *relation* between sets X and Y is a subset $\mathcal{R} \subseteq X \times Y$. We write $\mathcal{R}(x, y)$ if $(x, y) \in \mathcal{R}$. For $x \in X$ and $A \subseteq X$ we put

$$\mathcal{R}[x] = \{ y \in Y : \mathcal{R}(x, y) \} \quad \text{and} \quad \mathcal{R}[A] = \bigcup_{x \in A} \mathcal{R}[x].$$

The *converse* of \mathcal{R} is the relation $\check{\mathcal{R}} \subseteq Y \times X$ (alternatively \mathcal{R}^\smile) where $\check{\mathcal{R}}(y, x)$ iff $\mathcal{R}(x, y)$ for $x \in X$ and $y \in Y$. The *composite* of $\mathcal{R} \subseteq X \times Y$ and $\mathcal{S} \subseteq Y \times Z$ is the relation $\mathcal{R}; \mathcal{S} \subseteq X \times Z$ where $\mathcal{R}; \mathcal{S}(x, z)$ iff there exists $y \in Y$ with $\mathcal{R}(x, y)$ and $\mathcal{S}(y, z)$. Let **Rel** denote the category whose objects are sets and whose morphisms are relations with the above composition. The identity morphism on X is the identity relation $\text{id}_X \subseteq X \times X$ with $\text{id}_X(x, y)$ iff $x = y$.

A *biclique* of a relation $\mathcal{R} \subseteq X \times Y$ is subset of the form $B_1 \times B_2 \subseteq \mathcal{R}$, where $B_1 \subseteq X$ and $B_2 \subseteq Y$. A set \mathcal{C} of bicliques forms a *biclique cover* if $\mathcal{R} = \bigcup \mathcal{C}$. The *bipartite dimension* of \mathcal{R} , denoted $\text{dim}(\mathcal{R})$, is the minimum cardinality of any biclique cover.

Languages. Let Σ^* be the set of finite words over an alphabet Σ including the empty word ε . A *language* is a subset L of Σ^* . We let $\bar{L} = \Sigma^* \setminus L$ denote the *complement* and $L^r = \{w^r : w \in L\}$ the *reverse* of L , where $\varepsilon^r = \varepsilon$ and $w^r = a_n \dots a_1$ for $w = a_1 \dots a_n$. The *left derivatives* and *two-sided derivatives* of L are, respectively, given by $u^{-1}L = \{w \in \Sigma^* : uw \in L\}$ and $u^{-1}Lv^{-1} = \{w \in \Sigma^* : uww \in L\}$ for $u, v \in \Sigma^*$; moreover for $U \subseteq \Sigma^*$ put $U^{-1}L = \bigcup_{u \in U} u^{-1}L$. For each fixed $L \subseteq \Sigma^*$, the following sets of languages will play a prominent role:

$$\text{LD}(L) \subseteq \text{SLD}(L) \subseteq \text{BLD}(L) \subseteq \text{BLRD}(L)$$

where $\text{LD}(L) = \{u^{-1}L : u \in \Sigma^*\}$ is the set of left derivatives, and $\text{SLD}(L)$, $\text{BLD}(L)$, $\text{BLRD}(L)$ denote its closure under finite unions, all set-theoretic boolean operations, and all set-theoretic boolean operations and two-sided derivatives, respectively. The final three form \cup -semilattices, and the final two are boolean algebras w.r.t. the set-theoretic operations.

A language L is *regular* if $\text{LD}(L)$ is a finite set; then the other three sets are finite too. The finite semilattices $\text{SLD}(L)$ and $\text{SLD}(L')$ are related by the fundamental isomorphism

$$\text{dr}_L : [\text{SLD}(L')]^{\text{op}} \xrightarrow{\cong} \text{SLD}(L), \quad K \mapsto (\overline{K^r})^{-1}L, \quad (2.1)$$

see [26, Proposition 3.13]. Equivalently, the map dr_L sends $V^{-1}L' \in \text{SLD}(L')$ to the largest element of $\text{SLD}(L)$ disjoint from V^r . It is closely connected to the *dependency relation* of L ,

$$\mathcal{DR}_L \subseteq \text{LD}(L) \times \text{LD}(L'), \quad \mathcal{DR}_L(u^{-1}L, v^{-1}L') : \iff uv^r \in L \quad \text{for } u, v \in \Sigma^*. \quad (2.2)$$

In fact, by [26, Theorem 3.15] we have

$$\mathcal{DR}_L(u^{-1}L, v^{-1}L') \iff u^{-1}L \not\subseteq \text{dr}_L(v^{-1}L') \quad \text{for } u, v \in \Sigma^*. \quad (2.3)$$

Since the boolean algebra $\text{BLD}(L)$ is generated by the left derivatives of L , its atoms (= join-irreducibles) are the congruence classes of the *Nerode left congruence* $\sim_L \subseteq \Sigma^* \times \Sigma^*$,

$$u \sim_L v \quad \text{iff} \quad \forall x \in \Sigma^* : u \in x^{-1}L \iff v \in x^{-1}L \quad \text{iff} \quad (u^r)^{-1}L' = (v^r)^{-1}L'. \quad (2.4)$$

Note that this relation is left-invariant, i.e. $u \sim_L v$ implies $wu \sim_L wv$ for all $w \in \Sigma^*$.

Similarly, the atoms of $\text{BLRD}(L)$ are the congruence classes of the *syntactic congruence* $\equiv_L \subseteq \Sigma^* \times \Sigma^*$, i.e. the monoid congruence on the free monoid Σ^* defined by

$$u \equiv_L v \quad \text{iff} \quad \forall x, y \in \Sigma^* : u \in x^{-1}Ly^{-1} \iff v \in x^{-1}Ly^{-1}. \quad (2.5)$$

The quotient monoid $\text{syn}(L) = \Sigma^*/\equiv_L$ is called the *syntactic monoid* of L , and the canonical map $\mu_L : \Sigma^* \rightarrow \text{syn}(L)$ sending $u \in \Sigma^*$ to its congruence class $[u]_{\equiv_L}$ is the *syntactic morphism*.

Automata. Fix a finite alphabet Σ . A *nondeterministic finite automaton* (a.k.a. *nfa*) $N = (Q, \delta, I, F)$ consists of a finite set Q (the states), relations $\delta = (\delta_a \subseteq Q \times Q)_{a \in \Sigma}$ (the transitions), and sets $I, F \subseteq Q$ (the initial states and final states). We write $q_1 \xrightarrow{a} q_2$ whenever $q_2 \in \delta_a[q_1]$. The language $L(N, q)$ *accepted* by a state $q \in Q$ consists of all words $w \in \Sigma^*$ such that $\delta_w[q] \cap F \neq \emptyset$, where $\delta_w \subseteq Q \times Q$ is the extended transition relation $\delta_{a_1}; \dots; \delta_{a_n}$ for $w = a_1 \dots a_n$ and $\delta_\varepsilon = \text{id}_Q$. The language *accepted* by N is defined $L(N) = \bigcup_{i \in I} L(N, i)$.

An nfa N is a *deterministic finite automaton* (a.k.a. *dfa*) if $I = \{q_0\}$ is a singleton set and each transition relation is a function $\delta_a : Q \rightarrow Q$. A dfa is a **JSL-dfa** if Q is a finite semilattice, each $\delta_a : Q \rightarrow Q$ is a semilattice morphism, and $F \subseteq Q$ forms a prime filter. It is often useful to represent a **JSL-dfa** in terms of morphisms

$$\mathfrak{2} \xrightarrow{i} Q \xrightarrow{\delta_a} Q \xrightarrow{f} \mathfrak{2}$$

where i is the unique morphism with $i(1) = q_0$ and f is given by $f(q) = 1$ iff $q \in F$. A **JSL-dfa morphism** from $A = (Q, \delta, i, f)$ to $A' = (Q', \delta', i', f')$ is a **JSL_f-morphism** $h : Q \rightarrow Q'$ preserving transitions via $h \circ \delta_a = \delta'_a \circ h$, preserving the initial state via $i' = h \circ i$, and both preserving and reflecting the final states via $f = f' \circ h$. Equivalently, h is a dfa morphism that is also a semilattice morphism, so in particular $L(A) = L(A')$. If Q is a subsemilattice of Q' and $h : Q \rightarrow Q'$ is the inclusion map, then A is called a *sub JSL-dfa* of A' .

Fix a regular language L . Viewed as a \cup -semilattice, $\text{BLRD}(L)$ carries the structure of a **JSL**-dfa with transitions $K \xrightarrow{a} a^{-1}K$, initial state L , and finals $\{K : \varepsilon \in K\}$. This restricts to sub **JSL**-dfas $\text{BLD}(L)$ and $\text{SLD}(L)$. Moreover $\text{LD}(L)$ forms a sub-dfa of $\text{SLD}(L)$, well-known [5] to be the *state-minimal dfa* for L , so we denote it by $\text{dfa}(L)$. The syntactic monoid $\text{syn}(L)$ is isomorphic to the *transition monoid* of $\text{dfa}(L)$, i.e. the monoid of all extended transition maps $\delta_w : \text{LD}(L) \rightarrow \text{LD}(L)$ ($w \in \Sigma^*$) with multiplication given by composition [27].

Analogously $\text{SLD}(L)$ is the *state-minimal JSL-dfa* for L . Up to isomorphism, it is the unique **JSL**-dfa for L that is **JSL-reachable** (i.e. every state is a join of states reachable from the initial state via transitions) and *simple* (i.e. distinct states accept distinct languages).

Nfas, dfas and **JSL**-dfas are expressively equivalent and accept precisely the regular languages. In particular, to every **JSL**-dfa $A = (Q, \delta, q_0, F)$ one can associate an equivalent nfa $J(A)$, the *nfa of join-irreducibles* [1, 2, 25]. Its states are given by the set $J(Q)$ of join-irreducibles of Q ; for any $q_1, q_2 \in J(Q)$ and $a \in \Sigma$ there is a transition $q_1 \xrightarrow{a} q_2$ in $J(A)$ iff $q_2 \leq_Q \delta_a(q_1)$; a state $q \in J(Q)$ is initial iff $q \leq_S q_0$, and final iff $q \in F$. For any $q \in J(Q)$, we have $L(A, q) = L(J(A), q)$. The *canonical residual finite state automaton* [7] for a regular language L is given by $N_L = J(\text{SLD}(L))$, the nfa of join-irreducibles of its minimal **JSL**-dfa.

Atomic and subatomic nfas. An nfa accepting the language $L \subseteq \Sigma^*$ is called *atomic* [4] if each state accepts a language from $\text{BLD}(L)$, and *subatomic* [26] if each state accepts a language from $\text{BLRD}(L)$. The *nondeterministic atomic complexity* $\text{natm}(L)$ of a regular language L is the least number of states of any atomic nfa accepting L . The *nondeterministic syntactic complexity* $\text{nsyn}(L)$ is the least number of states of any subatomic nfa accepting L . Subatomic nfas are intimately connected to syntactic monoids: the atoms of $\text{BLRD}(L)$ are the elements of $\text{syn}(L)$, so an nfa accepting L is subatomic iff its individual states accept unions of syntactic congruence classes. Additionally $\text{nsyn}(L)$ can be characterized via *boolean representations* of $\text{syn}(L)$, i.e. monoid morphisms $\varrho : \text{syn}(L) \rightarrow \mathbf{JSL}_f(S, S)$ into the endomorphisms of a finite semilattice [26]. For a detailed exposition we refer to *op. cit.*

These complexity measures are related to the *nondeterministic state complexity* $\text{ns}(L)$, i.e. the least number of states of any (unrestricted) nfa accepting L . In particular,

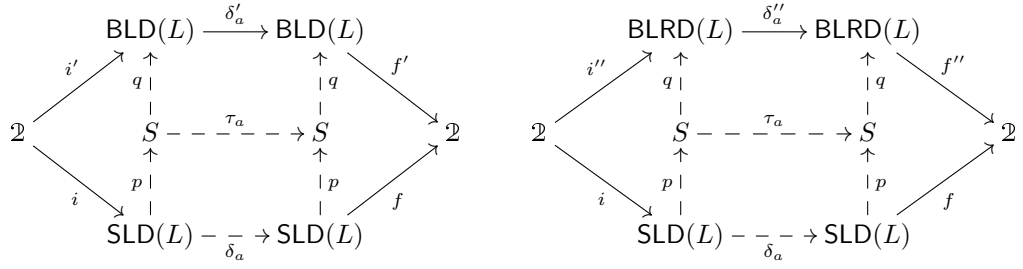
$$\dim(\mathcal{DR}_L) \leq \text{ns}(L) \leq \text{nsyn}(L) \leq \text{natm}(L). \quad (2.6)$$

The first inequality is due to Gruber and Holzer [10] (see also [26, Theorem 4.8] for a purely algebraic proof), while the others arise by restricting admissible nondeterministic acceptors.

Importantly, small atomic and subatomic nfas can be characterized in terms of **JSL**-dfas. The following theorem involves two commuting diagrams of semilattice morphisms, whose lower and upper paths are the canonical **JSL**-dfas described earlier.

► **Theorem 2.1.** *Let $L \subseteq \Sigma^*$ be a regular language.*

1. $\text{natm}(L)$ is the least number k such that there exists a finite semilattice S with $|J(S)| \leq k$ and **JSL**_f-morphisms p, q and τ_a ($a \in \Sigma$) making the left-hand diagram below commute.
2. $\text{nsyn}(L)$ is the least number k such that there exists a finite semilattice S with $|J(S)| \leq k$ and **JSL**_f-morphisms p, q and τ_a ($a \in \Sigma$) making the right-hand diagram below commute.



Proof. We only prove part (1), the proof of (2) being completely analogous.

Suppose there exists a finite semilattice S with $|J(S)| = k$ and \mathbf{JSL}_f -morphisms p, q and $(\tau_a)_{a \in \Sigma}$ making the left diagram commute. Then $A = (S, \tau, p \circ i, f' \circ q)$ is a \mathbf{JSL} -dfa and $p: \mathbf{SLD}(L) \rightarrow A$ and $q: A \rightarrow \mathbf{BLD}(L)$ are \mathbf{JSL} -dfa morphisms. Since \mathbf{JSL} -dfa morphisms preserve the accepted language, and every state $K \in \mathbf{BLD}(L)$ accepts the language K , it follows that A accepts L and every state of A accepts a language from $\mathbf{BLD}(L)$. Thus the nfa $J(A)$ of join-irreducibles corresponding to A is an atomic nfa for L with k states.

Conversely, assume $N = (Q, \delta, I, F)$ is a k -state atomic nfa accepting L . Form the \cup -semilattice $S = \text{langs}(N)$ of all languages $L(N, X)$ accepted by subsets $X \subseteq Q$. Note that $\mathbf{SLD}(L) \subseteq S \subseteq \mathbf{BLD}(L)$: the first inclusion holds because $u^{-1}L = L(N, \delta_w[I]) \in S$ for every $u \in \Sigma^*$, and the second one because N is atomic. We define the semilattice endomorphisms

$$\tau_a: S \rightarrow S \quad \text{by} \quad \tau_a(K) = a^{-1}K \quad \text{for } K \in S,$$

Letting $p: \mathbf{SLD}(L) \rightarrow S$ and $q: S \rightarrow \mathbf{BLD}(L)$ denote the inclusions, the left diagram commutes. Moreover $|J(S)| \leq k$ since S is join-generated by the elements $L(N, q)$ for $q \in Q$. \blacktriangleleft

3 Representing Finite Semilattices as Finite Relations

We have seen that atomic and subatomic nfes amount to certain dfes with semilattice structure. To obtain our NP-completeness results concerning the computation of small (sub)atomic acceptors we will study succinct representations of the corresponding \mathbf{JSL}_f -diagrams from Theorem 2.1. For this purpose, we start with the following key observation:

Any finite semilattice S is completely determined by its *poset of irreducibles* [23], i.e. the relation $\preceq_S \subseteq J(S) \times M(S)$ between join-irreducibles and meet-irreducibles.

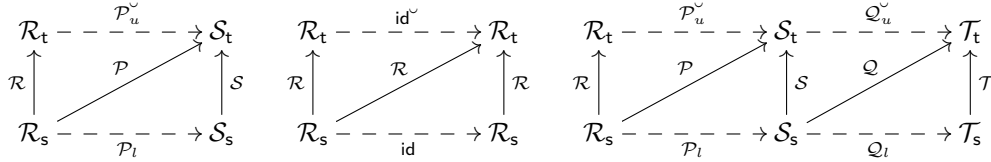
We now prove that this extends to an *equivalence* between the category \mathbf{JSL}_f of finite semilattices and another category called \mathbf{Dep} . Its objects are the relations between finite sets and its morphisms represent semilattice morphisms as relations. The equivalence is inspired by Moshier's *categories of contexts* [16, 24] and will serve as the conceptual basis of our work.

► **Definition 3.1** (The category of dependency relations). The objects of the category \mathbf{Dep} are the relations $\mathcal{R} \subseteq \mathcal{R}_s \times \mathcal{R}_t$ between finite sets. Far less obviously,

a morphism $\mathcal{P}: \mathcal{R} \rightarrow \mathcal{S}$ is a relation $\mathcal{P} \subseteq \mathcal{R}_s \times \mathcal{S}_t$ that factorizes through \mathcal{R} and \mathcal{S} , i.e. the left \mathbf{Rel} -diagram below commutes for some $\mathcal{P}_l \subseteq \mathcal{R}_s \times \mathcal{S}_s$ and $\mathcal{P}_u \subseteq \mathcal{S}_t \times \mathcal{R}_t$.

The identity morphism for \mathcal{R} is $\text{id}_{\mathcal{R}} = \mathcal{R}$, see the central diagram below. The composite $\mathcal{P}; \mathcal{Q}: \mathcal{R} \rightarrow \mathcal{T}$ of $\mathcal{P}: \mathcal{R} \rightarrow \mathcal{S}$ and $\mathcal{Q}: \mathcal{S} \rightarrow \mathcal{T}$ is any of the five equivalent relational compositions

starting from the bottom left corner and ending at the top right corner of the rightmost diagram below; that is, $\mathcal{P} \circledast \mathcal{Q} := \mathcal{P}_l; \mathcal{Q}_l; \mathcal{T} = \mathcal{P}_l; \mathcal{Q} = \mathcal{P}_l; \mathcal{S}; \mathcal{Q}_u = \mathcal{P}; \mathcal{Q}_u = \mathcal{R}; \mathcal{P}_u; \mathcal{Q}_u$. (Note that we use the symbol \circledast for composition in **Dep** and $;$ for composition in **Rel**, and recall that $(-)^{\check{}}$ denotes the converse relation.)



One readily verifies that **Dep** is a well-defined category; in particular, the composition is independent of the choice of the lower and upper witnesses $(-)_l$ and $(-)_u$.

► **Remark 3.2.**

1. Using the converse upper witness may seem strange. Although technically unnecessary, it fits the self-duality of **Dep** taking the converse on objects and morphisms. Moreover $f; \check{\mathcal{L}}_T = \check{\mathcal{L}}_S; f_*$ for any **JSL_f**-morphism $f: S \rightarrow T$ via the adjoint relationship; that is, f induces a **Dep**-morphism from $\check{\mathcal{L}}_S$ to $\check{\mathcal{L}}_T$ with lower witness f and upper witness f_* .
2. The witnesses of a **Dep**-morphism $\mathcal{P}: \mathcal{R} \rightarrow \mathcal{S}$ are closed under unions. The maximal lower witness $\mathcal{P}_- \subseteq \mathcal{R}_s \times \mathcal{S}_s$ is given by

$$\mathcal{P}_-(x, y) : \iff S[y] \subseteq \mathcal{P}[x] \quad \text{for } x \in \mathcal{R}_s, y \in \mathcal{S}_s,$$

and the maximal upper witness $\mathcal{P}_+ \subseteq \mathcal{S}_t \times \mathcal{R}_t$ by

$$\mathcal{P}_+(y, x) : \iff \check{\mathcal{R}}[x] \subseteq \check{\mathcal{P}}[y] \quad \text{for } x \in \mathcal{R}_t, y \in \mathcal{S}_t.$$

► **Theorem 3.3** (Fundamental equivalence). *The categories **JSL_f** and **Dep** are equivalent.*

1. The equivalence functor $\text{Pirr}: \mathbf{JSL}_f \rightarrow \mathbf{Dep}$ maps a finite semilattice S to the **Dep**-object

$$\text{Pirr}(S) := \check{\mathcal{L}}_S \subseteq J(S) \times M(S),$$

and a **JSL_f**-morphism $f: S \rightarrow T$ to the **Dep**-morphism

$$\text{Pirr}(f): \text{Pirr}(S) \rightarrow \text{Pirr}(T), \quad \text{Pirr}(f)(j, m) : \iff f(j) \check{\mathcal{L}}_T m \quad \text{for } j \in J(S), m \in M(T).$$

2. The inverse $\text{Open}: \mathbf{Dep} \rightarrow \mathbf{JSL}_f$ maps a **Dep**-object \mathcal{R} to its semilattice of open sets

$$\text{Open}(\mathcal{R}) := (\{\mathcal{R}[X] : X \subseteq \mathcal{R}_s\}, \subseteq),$$

and a **Dep**-morphism $\mathcal{P}: \mathcal{R} \rightarrow \mathcal{S}$ to the **JSL_f**-morphism

$$\text{Open}(\mathcal{P}): \text{Open}(\mathcal{R}) \rightarrow \text{Open}(\mathcal{S}), \quad \text{Open}(\mathcal{P})(O) := \mathcal{P}_+[O] \quad \text{for } O \in \text{Open}(\mathcal{R}),$$

where $\mathcal{P}_+ \subseteq \mathcal{S}_t \times \mathcal{R}_t$ is the maximal upper witness of \mathcal{P} .

► **Remark 3.4.** In the definition of $\text{Pirr}(S)$ one may replace $J(S)$ and $M(S)$ by any two sets $J, M \subseteq S$ of join- and meet-generators modulo **Dep**-isomorphism. Indeed, since the equivalence functor Open reflects isomorphisms, this follows immediately from the **JSL_f**-isomorphism $\text{Open}(\check{\mathcal{L}}_S \cap J \times M) \cong \text{Open}(\check{\mathcal{L}}_S \cap J(S) \times M(S))$ given by $O \mapsto O \cap M(S)$.

► **Remark 3.5.** Bijectively relabeling the domain and codomain of a relation defines a **Dep**-isomorphism, the witnesses being the relabelings.

We now show that for every regular language L , the semilattices $\text{SLD}(L)$, $\text{BLD}(L)$ and $\text{BLRD}(L)$ equipped with their canonical **JSL**-dfa structure (see Section 2) translate under the equivalence functor Pirr into familiar concepts from automata theory. The translations are summarized in Table 1 and explained in Examples 3.6–3.8 below.

■ **Table 1** Canonical **JSL**-dfas and their corresponding **Dep**-structures.

JSL_f			Dep		
$\mathbb{2} \xrightarrow{i} \text{SLD}(L) \xrightarrow{\delta_a} \text{SLD}(L) \xrightarrow{f} \mathbb{2}$	$\xrightarrow{\delta_a}$	\xrightarrow{f}	$\text{id}_1 \xrightarrow{\mathcal{I}} \mathcal{DR}_L \xrightarrow{\mathcal{DR}_{L,a}} \mathcal{DR}_L \xrightarrow{\mathcal{F}} \text{id}_1$	$\xrightarrow{\mathcal{DR}_{L,a}}$	$\xrightarrow{\mathcal{F}}$
$\mathbb{2} \xrightarrow{i'} \text{BLD}(L) \xrightarrow{\delta'_a} \text{BLD}(L) \xrightarrow{f'} \mathbb{2}$	$\xrightarrow{\delta'_a}$	$\xrightarrow{f'}$	$\text{id}_1 \xrightarrow{\mathcal{I}'} \text{id}_{\Sigma^*/\sim_L} \xrightarrow{\mathcal{D}'_a} \text{id}_{\Sigma^*/\sim_L} \xrightarrow{\mathcal{F}'} \text{id}_1$	$\xrightarrow{\mathcal{D}'_a}$	$\xrightarrow{\mathcal{F}'}$
$\mathbb{2} \xrightarrow{i''} \text{BLRD}(L) \xrightarrow{\delta''_a} \text{BLRD}(L) \xrightarrow{f''} \mathbb{2}$	$\xrightarrow{\delta''_a}$	$\xrightarrow{f''}$	$\text{id}_1 \xrightarrow{\mathcal{I}''} \text{id}_{\text{syn}(L)} \xrightarrow{\mathcal{D}''_a} \text{id}_{\text{syn}(L)} \xrightarrow{\mathcal{F}''} \text{id}_1$	$\xrightarrow{\mathcal{D}''_a}$	$\xrightarrow{\mathcal{F}''}$

► **Example 3.6** (State-minimal **JSL**-dfa vs. dependency relation \mathcal{DR}_L). Let us start with the observation that $\text{SLD}(L)$ is join-generated by $\text{LD}(L)$ and meet-generated by $\text{dr}_L[\text{LD}(L')]$. The latter follows via the fundamental isomorphism (2.1). Then

$$\text{Pirr}(\text{SLD}(L))(u^{-1}L, \text{dr}_L(v^{-1}L')) \stackrel{\text{def.}}{\iff} u^{-1}L \not\subseteq \text{dr}_L(v^{-1}L') \stackrel{(2.3)}{\iff} \mathcal{DR}_L(u^{-1}L, v^{-1}L')$$

for every $u^{-1}L \in J(\text{SLD}(L))$ and $v^{-1}L' \in J(\text{SLD}(L'))$. Thus,

$$\text{Pirr}(\text{SLD}(L)) \text{ is a bijective relabeling of } \mathcal{DR}_L \text{ restricted to } J(\text{SLD}(L)) \times J(\text{SLD}(L')).$$

By Remark 3.4 we know $\text{Pirr}(\text{SLD}(L))$ is isomorphic to the domain-codomain extension $\not\subseteq \subseteq \text{LD}(L) \times \text{dr}_L[\text{LD}(L')]$ and thus also to the dependency relation \mathcal{DR}_L by Remark 3.5. Then the **JSL**-dfa structure of the semilattice $\text{SLD}(L)$ translates into the category of dependency relations as shown in Table 1, where id_1 is the identity relation on $1 = \{*\}$ and

$$\begin{aligned} \mathcal{I} &\subseteq 1 \times \text{LD}(L'), & \mathcal{DR}_{L,a} &\subseteq \text{LD}(L) \times \text{LD}(L'), & \mathcal{F} &\subseteq \text{LD}(L) \times 1, \\ \mathcal{I}(*, v^{-1}L') &\Leftrightarrow v \in L', & \mathcal{DR}_{L,a}(u^{-1}L, v^{-1}L') &\Leftrightarrow uav' \in L, & \mathcal{F}(u^{-1}L, *) &\Leftrightarrow u \in L. \end{aligned}$$

► **Example 3.7** ($\text{BLD}(L)$ vs. the Nerode left congruence \sim_L). In Section 2 we observed that the atoms of the boolean algebra $\text{BLD}(L)$ are the congruence classes of the Nerode left congruence. Then the co-atoms are their relative complements, and

$$\text{Pirr}(\text{BLD}(L))([u]_{\sim_L}, \overline{[v]_{\sim_L}}) \stackrel{\text{def.}}{\iff} [u]_{\sim_L} \not\subseteq \overline{[v]_{\sim_L}} \iff [u]_{\sim_L} = [v]_{\sim_L}.$$

By Remark 3.5, we see that $\text{BLD}(L)$ corresponds to the **Dep**-object $\text{id}_{\Sigma^*/\sim_L}$, and its **JSL**-dfa structure translates into the category of dependency relations as indicated in Table 1, where

$$\begin{aligned} \mathcal{I}' &\subseteq 1 \times \Sigma^*/\sim_L, & \mathcal{D}'_a &\subseteq \Sigma^*/\sim_L \times \Sigma^*/\sim_L, & \mathcal{F}' &\subseteq \Sigma^*/\sim_L \times 1, \\ \mathcal{I}'(*, [u]_{\sim_L}) &\Leftrightarrow u \in L, & \mathcal{D}'_a([u]_{\sim_L}, [v]_{\sim_L}) &\Leftrightarrow [v]_{\sim_L} \subseteq a^{-1}[u]_{\sim_L}, & \mathcal{F}'([u]_{\sim_L}, *) &\Leftrightarrow u \sim_L \varepsilon. \end{aligned}$$

We note that the above relations induce an nfa

$$(\Sigma^*/\sim_L, (\mathcal{D}'_a)_{a \in \Sigma}, \mathcal{I}'[*], \check{\mathcal{F}}'[*]) \quad \text{known as the } \acute{a}tomaton \text{ for the language } L [4].$$

► **Example 3.8** ($\text{BLRD}(L)$ vs. the syntactic monoid $\text{syn}(L)$). Analogously, the boolean algebra $\text{BLRD}(L)$ corresponds to the **Dep**-object $\text{id}_{\text{syn}(L)}$. Its semilattice dfa structure translates into the category of dependency relations as shown in Table 1, where

$$\begin{aligned} \mathcal{I}'' &\subseteq 1 \times \text{syn}(L), & \mathcal{D}''_a &\subseteq \text{syn}(L) \times \text{syn}(L), & \mathcal{F}'' &\subseteq \text{syn}(L) \times 1, \\ \mathcal{I}''(*, [u]_{\equiv_L}) &\Leftrightarrow u \in L, & \mathcal{D}''_a([u]_{\equiv_L}, [v]_{\equiv_L}) &\Leftrightarrow [v]_{\equiv_L} \subseteq a^{-1}[u]_{\equiv_L}, & \mathcal{F}''([u]_{\equiv_L}, *) &\Leftrightarrow u \equiv_L \varepsilon. \end{aligned}$$

We conclude this section with two lemmas establishing important properties of the equivalence. The first concerns the bipartite dimension of relations (see Section 2):

► **Lemma 3.9.** *Let \mathcal{R} be a relation between finite sets.*

1. $\dim(\mathcal{R})$ is the least $|J(S)|$ of any injective \mathbf{JSL}_f -morphism $m: \text{Open}(\mathcal{R}) \rightarrow S$.
 2. $\dim(\mathcal{R})$ is invariant under isomorphism, i.e. $\mathcal{R} \cong \mathcal{S}$ in \mathbf{Dep} implies $\dim(\mathcal{R}) = \dim(\mathcal{S})$.
- The second explicitly describes the join- and meet-irreducibles of the semilattice $\text{Open}(\mathcal{R})$.

► **Notation 3.10.** For $\mathcal{R} \subseteq \mathcal{R}_s \times \mathcal{R}_t$ we define the following operator on the power set of \mathcal{R}_t :

$$\text{in}_{\mathcal{R}}: \mathcal{P}(\mathcal{R}_t) \rightarrow \mathcal{P}(\mathcal{R}_t), \quad Y \mapsto \bigcup \{ \mathcal{R}[X] : X \subseteq \mathcal{R}_s \text{ and } \mathcal{R}[X] \subseteq Y \}.$$

Thus, $\text{in}_{\mathcal{R}}(Y)$ is the largest open set of \mathcal{R} contained in $Y \subseteq \mathcal{R}_t$.

► **Lemma 3.11.** *Let $\mathcal{R} \subseteq \mathcal{R}_s \times \mathcal{R}_t$ be a relation between finite sets.*

1. $J(\text{Open}(\mathcal{R}))$ consists of all sets $\mathcal{R}[x]$ ($x \in \mathcal{R}_s$) that cannot be expressed as a union of smaller such sets, i.e. $\mathcal{R}[x] = \bigcup_{i \in I} \mathcal{R}[x_i]$ implies $\mathcal{R}[x] = \mathcal{R}[x_i]$ for some $i \in I$.
2. $M(\text{Open}(\mathcal{R}))$ consists of all sets $\text{in}_{\mathcal{R}}(\mathcal{R}_t \setminus \{y\})$ such that $\check{\mathcal{R}}[y]$ lies in $J(\text{Open}(\check{\mathcal{R}}))$.

4 Nuclear Languages and Lattice Languages

As a further technical tool, we now introduce two classes of regular languages. They are well-behaved w.r.t. their small nfas and will emerge at the heart of our NP-completeness proofs in Section 5. Their definition rests on the notion of a *nuclear morphism* in \mathbf{JSL}_f , originating from the theory of symmetric monoidal closed categories [12, 28]. Recall that a finite semilattice is a *distributive lattice* if $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ for all elements x, y, z .

► **Definition 4.1** (Nuclear language). A \mathbf{JSL}_f -morphism $f: S \rightarrow T$ is *nuclear* if it factorizes through a finite distributive lattice, i.e. $f = (S \xrightarrow{g} D \xrightarrow{h} T)$ for some finite distributive lattice D and \mathbf{JSL}_f -morphisms g, h . A regular language $L \subseteq \Sigma^*$ is *nuclear* if the transition morphisms $\delta_a = a^{-1}(-): \text{SLD}(L) \rightarrow \text{SLD}(L)$ ($a \in \Sigma$) of its minimal \mathbf{JSL} -dfa are nuclear.

► **Example 4.2** (BiRFSAs). A regular language L is *biRFSAs* [19] if $(N_L)^r \cong N_{L^r}$, that is, the canonical residual finite state automata for L and L^r (see Section 2) are reverse-isomorphic. In [26, Example 5.7] we proved that the biRFSAs languages are precisely those whose semilattice $\text{SLD}(L)$ is distributive. Thus biRFSAs languages are nuclear.

There is a natural subclass of nuclear languages which need not be biRFSAs:

► **Definition 4.3** (Lattice language). For any $S \in \mathbf{JSL}_f$ we define the language $L(S) \subseteq \Sigma^*$,

$$\Sigma := \{ \langle j \rangle : j \in J(S) \} \cup \{ |m\rangle : m \in M(S) \} \quad \text{and} \quad L(S) := \bigcap_{j \leq_S m} \overline{\Sigma^* \langle j \rangle |m\rangle \Sigma^*}.$$

Then Σ is the disjoint union of $J(S)$ and $M(S)$ (with the notation $\langle j \rangle$ and $|m\rangle$ used to distinguish between elements of the two summands), and $L(S)$ consists of all words over Σ not containing any factor $\langle j \rangle |m\rangle$ with $j \leq_S m$.

► **Lemma 4.4.** *For any $S \in \mathbf{JSL}_f$, the language $L(S)$ is nuclear and $S \cong \text{SLD}(L(S))$.*

Crucially, for nuclear and lattice languages some of the relations (2.6) hold with equality:

► **Proposition 4.5.**

1. If L is a nuclear language then $\text{ns}(L) = \dim(\mathcal{DR}_L)$.
2. If $L = L(S)$ is a lattice language then $\text{natm}(L) = \text{nsyn}(L) = \text{ns}(L) = \dim(\mathcal{DR}_L)$.

These equalities are the key fact making our reductions in the next section work.

5 Complexity of Computing Small (Sub)Atomic Acceptors

We are ready to present our main complexity results on small (sub)atomic nfes. First we consider the slightly simpler atomic case, phrased as the following decision problem:

DFA + DFA^r → NFA_{atm}

Input: Two dfas A and B such that $L(A) = L(B)^r$ and a natural number k .

Task: Decide whether there exists a k -state atomic nfa equivalent to A , i.e. $\text{natm}(L(A)) \leq k$.

► **Remark 5.1.** Taking mutually reverse dfas (A, B) as input permits an efficient computation of the dependency relation $\mathcal{DR}_L \subseteq \text{LD}(L) \times \text{LD}(L^r)$ of $L = L(A)$. One may assume A and B are minimal dfas, so that their state sets Q_A and Q_B are in bijective correspondence with $\text{LD}(L)$ and $\text{LD}(L^r)$. For $p \in Q_A$ choose some $w_A(p) \in \Sigma^*$ sending the initial state to p ; analogously choose $w_B(q) \in \Sigma^*$ for $q \in Q_B$. Then \mathcal{DR}_L is a bijective relabeling of

$$\widetilde{\mathcal{DR}}_L \subseteq Q_A \times Q_B \quad \text{where} \quad \widetilde{\mathcal{DR}}_L(p, q) :\iff A \text{ accepts } w_A(p)w_B(q)^r,$$

so it is computable in polynomial time from A and B . A completely analogous argument applies to the relations \mathcal{I} , $\mathcal{DR}_{L,a}$ and \mathcal{F} from Example 3.6.

► **Theorem 5.2.** *The problem DFA + DFA^r → NFA_{atm} is NP-complete.*

We establish the upper and lower bound separately in the next two propositions. Both their proofs are based on the fundamental equivalence between **JSL_f** and **Dep**.

► **Proposition 5.3.** *The problem DFA + DFA^r → NFA_{atm} is in NP.*

Proof.

1. One can check in polynomial time whether a given pair (A, B) of dfas forms a valid input, i.e. satisfies $L(A) = L(B)^r$. In fact, this condition is equivalent to $\overline{L(A)} \cap L(B)^r = \overline{L(B)} \cap L(A)^r = \emptyset$. Using the standard methods for complementing dfas and reversing and intersecting nfes, one can construct nfes for $\overline{L(A)} \cap L(B)^r$ and $\overline{L(B)} \cap L(A)^r$ of size polynomial in $|A|$ and $|B|$, the number of states of A and B , and check for emptiness by verifying that no final state is reachable from the initial states.
2. Let A and B be dfas accepting the languages L and L^r , respectively, and let k be a natural number. We claim the following three statements to be equivalent:
 - a. There exists an atomic nfa accepting L with at most k states.
 - b. There exists a finite semilattice S with $|J(S)| \leq k$ and **JSL_f**-morphisms p, q and τ_a ($a \in \Sigma$) making the left diagram below commute.
 - c. There exists a **Dep**-object $\mathcal{S} \subseteq \mathcal{S}_s \times \mathcal{S}_t$ with $|\mathcal{S}_s| \leq k$ and $|\mathcal{S}_t| \leq |B|$ and **Dep**-morphisms \mathcal{P}, \mathcal{Q} and \mathcal{T}_a ($a \in \Sigma$) making the right diagram below commute (cf. Example 3.6/3.7).

$$\begin{array}{ccc}
 \text{BLD}(L) & \xrightarrow{\delta'_a} & \text{BLD}(L) \\
 \uparrow i' & & \uparrow f' \\
 \mathbb{2} & & \mathbb{2} \\
 \downarrow i & & \downarrow f \\
 \text{SLD}(L) & \xrightarrow{\delta_a} & \text{SLD}(L)
 \end{array}
 \quad
 \begin{array}{ccc}
 \text{id}_{\Sigma^*/\sim_L} & \xrightarrow{\mathcal{D}'_a} & \text{id}_{\Sigma^*/\sim_L} \\
 \uparrow \mathcal{I}' & & \uparrow \mathcal{F}' \\
 \text{id}_1 & & \text{id}_1 \\
 \downarrow \mathcal{I} & & \downarrow \mathcal{F} \\
 \mathcal{DR}_L & \xrightarrow{\mathcal{DR}_{L,a}} & \mathcal{DR}_L
 \end{array}
 \quad (5.1)$$

In fact, (a) \Leftrightarrow (b) was shown in Theorem 2.1(1), and (b) \Leftrightarrow (c) follows from the equivalence between \mathbf{JSL}_f and \mathbf{Dep} . To see this, note that in the left diagram we may assume q to be injective; otherwise, factorize q as $q = q' \circ e'$ with e surjective and q' injective and work with q' instead of q . By the self-duality of \mathbf{JSL}_f , dualizing q yields a surjective morphism from $\mathbf{BLD}(L) \cong \mathbf{BLD}(L)^{\text{op}}$ to S^{op} . Thus,

$$|M(S)| = |J(S^{\text{op}})| \leq |J(\mathbf{BLD}(L))| = |\Sigma^*/\sim_L| = |\mathbf{LD}(L^r)| \leq |B|.$$

In the two last steps, we use that the congruence classes of \sim_L correspond bijectively to left derivatives of L^r by (2.4), and that $\mathbf{LD}(L^r)$ is the set of states of the minimal dfa for L^r .

By Example 3.6 and 3.7 the upper and lower path of the left diagram in \mathbf{JSL}_f correspond under the equivalence functor Pirr to the upper and lower path of the right diagram in \mathbf{Dep} . Therefore, Theorem 3.3 shows the two diagrams to be equivalent.

3. From (a) \Leftrightarrow (c) we deduce that the relations \mathcal{S} , \mathcal{P} , \mathcal{Q} and \mathcal{T}_a ($a \in \Sigma$) constitute a short certificate for the existence of an atomic nfa for L with at most k states. Commutativity of the right diagram can be checked in polynomial time because all the relations appearing in the upper and lower path can be efficiently computed from the given dfas A and B . Indeed, for the lower path we have already noted this in Remark 5.1, and the upper path emerges from the minimal dfa for L^r , using that $\Sigma^*/\sim_L \cong \mathbf{LD}(L^r)$. \blacktriangleleft

► **Remark 5.4.** An alternative proof that $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}_{\text{atm}}$ is in NP uses the following characterization of atomic nfes. Given an nfa N , let $\text{rsc}(N^r)$ denote the dfa obtained by determinizing the reverse nfa N^r via the subset construction and restricting to its reachable part. Then N is atomic iff $\text{rsc}(N^r)$ is a minimal dfa [4, Corollary 2]. Thus, given a pair (A, B) of mutually reversed dfas, to decide whether $\text{natm}(L(A)) \leq k$ one may guess a k -state nfa N and verify that $\text{rsc}(N^r)$ is a minimal dfa equivalent to B . One advantage of our above categorical argument is that it yields simple certificates in the form of \mathbf{Dep} -morphisms subject to certain commutative diagrams, which amount to solutions of equations in \mathbf{Rel} . The latter may be directly computed using a SAT solver, leading to a practical approach to finding small atomic acceptors (cf. [9]). To this effect, let us note that the proof of Proposition 5.3 actually shows how to *construct* small atomic nfes rather than just deciding their existence: every certificate $\mathcal{S}, \mathcal{P}, \mathcal{Q}, \mathcal{T}_a$ ($a \in \Sigma$) yields an atomic nfa with states \mathcal{S}_s , transitions given by $(\mathcal{T}_a)_- \subseteq \mathcal{S}_s \times \mathcal{S}_s$ for $a \in \Sigma$, initial states $(\mathcal{I} \mathbin{\text{;}} \mathcal{P})_-[*] \subseteq \mathcal{S}_s$ and final states $(\mathcal{Q} \mathbin{\text{;}} \mathcal{F}')_-[*] \subseteq \mathcal{S}_s$. (Recall that $\mathbin{\text{;}}$ denotes composition in \mathbf{Dep} and $(-)_-$ denotes the maximum lower witness of a \mathbf{Dep} -morphism, see Remark 3.2.) In fact, this is precisely the nfa of join-irreducibles of the \mathbf{JSL} -dfa $(S, \tau, p \circ i, f' \circ q)$ induced by the left diagram in (5.1). Analogous reasoning also applies to the computation of small subatomic nfes treated in Theorem 5.7 below.

► **Proposition 5.5.** *The problem $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}_{\text{atm}}$ is NP-hard.*

Proof. We devise a polynomial-time reduction from the NP-complete problem **BICLIQUE COVER** [8]: given a pair (\mathcal{R}, k) of a relation $\mathcal{R} \subseteq \mathcal{R}_s \times \mathcal{R}_t$ between finite sets and a natural number k , decide whether \mathcal{R} has a biclique cover of size at most k , i.e. $\dim(\mathcal{R}) \leq k$.

For any (\mathcal{R}, k) , let $S = \text{Open}(\mathcal{R})$ be the finite semilattice of open sets corresponding to the \mathbf{Dep} -object \mathcal{R} , cf. Theorem 3.3, and let $L = L(S)$ be its lattice language. We claim that the desired reduction is given by

$$(\mathcal{R}, k) \longmapsto (\text{dfa}(L), \text{dfa}(L^r), k),$$

78:12 Syntactic Minimization of Nondeterministic Finite Automata

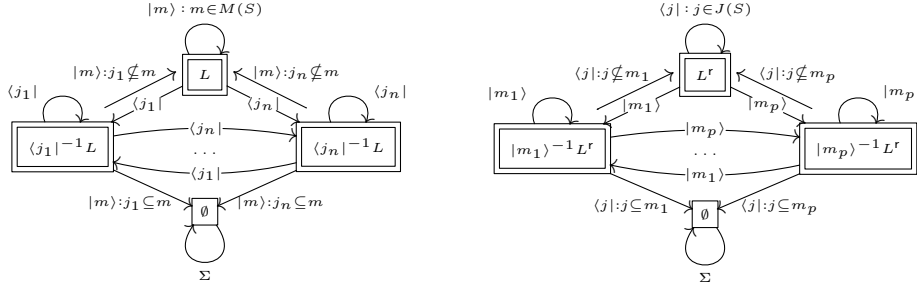
where $\text{dfa}(L)$ and $\text{dfa}(L')$ are the minimal dfas for L and L' . Thus, we need to prove that (a) $\dim(\mathcal{R}) = \text{natm}(L)$, and (b) the two dfas can be computed in polynomial time from \mathcal{R} .

Ad (a). We have the following sequence of **Dep**-isomorphisms:

$$\mathcal{R} \xleftarrow[\text{Thm 3.3}]{\cong} \text{Pirr}(\text{Open}(\mathcal{R})) = \text{Pirr}(S) \xleftarrow[\text{Lem 4.4}]{\cong} \text{Pirr}(\text{SLD}(L(S))) = \text{Pirr}(\text{SLD}(L)) \xleftarrow[\text{Ex 3.6}]{\cong} \mathcal{DR}_L.$$

Lemma 3.9(2) and Proposition 4.5 then imply $\dim(\mathcal{R}) = \dim(\mathcal{DR}_L) = \text{natm}(L)$.

Ad (b). Let $J(\text{Open}(\mathcal{R})) = \{j_1, \dots, j_n\}$ and $M(\text{Open}(\mathcal{R})) = \{m_1, \dots, m_p\}$. Then $\text{dfa}(L)$ and $\text{dfa}(L')$ are the automata depicted below, where L and L' are their respective initial states.



Both automata can be computed in polynomial time from \mathcal{R} using Lemma 3.11. \blacktriangleleft

Next, we turn to the computation of small subatomic nfas. While in the atomic case the input language was specified by a pair of dfas, we now assume an algebraic representation:

► **Definition 5.6.** A *monoid recognizer* is a triple (M, h, F) of a finite monoid M , a map $h: \Sigma \rightarrow M$ and a subset $F \subseteq M$. The language *recognized* by (M, h, F) is given by $L(M, h, F) = \bar{h}^{-1}[F]$, where $\bar{h}: \Sigma^* \rightarrow M$ is the unique extension of h to a monoid morphism.

It is well-known [27] that a language L is regular iff it has a monoid recognizer. In this case, a *minimal* monoid recognizer for L is given by $(\text{syn}(L), \mu_L, F_L)$ where $\mu_L: \Sigma \rightarrow \text{syn}(L)$ is the domain restriction of the syntactic morphism and $F_L = \{[w]_{\equiv_L} : w \in L\}$. It satisfies $|\text{syn}(L)| \leq |M|$ for every recognizer (M, h, F) of L . Consider the following decision problem:

MON \rightarrow **NFA**_{syn}

Input: A monoid recognizer (M, h, F) and a natural number k .

Task: Decide whether there exists a k -state subatomic nfa accepting $L(M, h, F)$.

Here we assume that the monoid M is explicitly given by its multiplication table.

► **Theorem 5.7.** *The problem **MON** \rightarrow **NFA**_{syn} is NP-complete.*

Proof sketch. The proof is conceptually similar to the one of Theorem 5.2. To show the problem to be in NP, one uses the algebraic characterization of $\text{nsyn}(L)$ in Theorem 2.1(2) and translates the ensuing **JSL**_f-diagram into **Dep**. To show NP-hardness, one reduces from **BICLIQUE COVER** via

$$(\mathcal{R}, k) \mapsto ((\text{syn}(L), \mu_L, F_L), k),$$

where again $L = L(\text{Open}(\mathcal{R}))$. \blacktriangleleft

Our complexity results indicate a trade-off, i.e. computing small subatomic nfas requires a less succinct representation of the input language. Generally, $|\text{dfa}(L)|, |\text{dfa}(L')| \leq |\text{syn}(L)|$ and the syntactic monoid can be far larger – even for nuclear languages.

► **Example 5.8.** For any natural number n consider the dfa $A_n = (\{0, \dots, n-1\}, \delta, 1, \{1\})$ over the alphabet $\Sigma = \{\pi, \tau\}$ with $\delta_\pi(i) = i + 1 \pmod n$ for $i = 0, \dots, n-1$, and $\delta_\tau(0) = 1$, $\delta_\tau(1) = 0$, $\delta_\tau(i) = i$ otherwise. Let $L_n = L(A_n)$ denote its accepted language. Then:

1. Both A_n and its reverse nfa are minimal dfas; in particular, $|\text{dfa}(L_n)| = |\text{dfa}(L_n^r)| = n$.
2. We have $|\text{syn}(L_n)| = n!$. To see this, recall that $\text{syn}(L_n)$ is the transition monoid of $A_n \cong \text{dfa}(L_n)$. It is generated by the n -cycle $\delta_\pi = (0\ 1\ \dots\ n-1)$ and the transposition $\delta_\tau = (0\ 1)$; then it equals the symmetric group S_n on n letters.
3. By part (1) the language L_n is *bideterministic* [30], i.e. accepted by a dfa whose reverse nfa is deterministic. This implies that the left derivatives of L_n are pairwise disjoint, so $\text{SLD}(L_n)$ is a boolean algebra. In particular, L_n is a nuclear language.

We finally further justify the inputs of $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}_{\text{atm}}$ and $\mathbf{MON} \rightarrow \mathbf{NFA}_{\text{syn}}$: the two modified problems $\mathbf{DFA} \rightarrow \mathbf{NFA}_{\text{atm}}$ and $\mathbf{DFA} \rightarrow \mathbf{NFA}_{\text{syn}}$ where only a (single) dfa is given are computationally much harder.

► **Theorem 5.9.** $\mathbf{DFA} \rightarrow \mathbf{NFA}_{\text{atm}}$ and $\mathbf{DFA} \rightarrow \mathbf{NFA}_{\text{syn}}$ are PSPACE-complete.

Proof. This follows by inspecting Jiang and Ravikumar's [15] argument that $\mathbf{DFA} \rightarrow \mathbf{NFA}$ is PSPACE-complete. These authors give a polynomial-time reduction from the PSPACE-complete problem **UNIVERSALITY OF MULTIPLE DFAS**, which asks whether a given list A_1, \dots, A_n of dfas over the same alphabet Σ satisfies $\bigcup_i L(A_i) = \Sigma^*$. For any A_1, \dots, A_n they construct a dfa A over some alphabet Γ and a natural number k such that:

1. If $\bigcup_i L(A_i) \neq \Sigma^*$, then every nfa accepting $L(A)$ requires at least $k+1$ states.
2. If $\bigcup_i L(A_i) = \Sigma^*$, then there exists an nfa accepting $L(A)$ with k states.

In the proof of (2), an explicit k -state nfa $N = (Q, \delta, \{q_0\}, F)$ with $L(N) = L(A)$ is given, see [15, Fig. 1]. It has the property that, after ε -elimination, for every state q there exists $w \in \Gamma^*$ with $\delta_w[q_0] = \{q\}$. This implies that every state q accepts a left derivative $w^{-1}L(N)$, i.e. N is a *residual* nfa [7]. In particular, N is both atomic and subatomic. Consequently, $(A_1, \dots, A_n) \mapsto (A, k)$ is also a reduction to both $\mathbf{DFA} \rightarrow \mathbf{NFA}_{\text{atm}}$ and $\mathbf{DFA} \rightarrow \mathbf{NFA}_{\text{syn}}$. ◀

6 Applications

We conclude this paper by outlining some useful consequences of our NP-completeness results concerning the computation of small nfes for specific classes of regular languages.

6.1 Nuclear Languages

As shown above, nuclear languages form a natural common generalization of bideterministic, biRFSA, and lattice languages. Let $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}$ be the variant of $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}_{\text{atm}}$ where the target nfes are arbitrary, i.e. the task is to decide $\text{ns}(L(A)) \leq k$. Then:

► **Theorem 6.1.** For nuclear languages, the problem $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}$ is NP-complete.

In fact, by Proposition 4.5(1) we have $\text{ns}(L) = \dim(\mathcal{DR}_L)$ for nuclear languages, so NP certificates are given by biclique covers. The NP-hardness proof is identical to the one of Theorem 5.2: the reduction involves a lattice language, which is nuclear by Lemma 4.4.

6.2 Unary languages

For unary regular languages $L \subseteq \{a\}^*$, every two-sided derivative $(a^i)^{-1}L(a^j)^{-1}$ is equal to the left derivative $(a^{i+j})^{-1}L$. Therefore, we have $\text{natm}(L) = \text{nsyn}(L)$ and the minimal dfa for L is the dfa structure of the syntactic monoid. From Theorem 5.7 we thus derive

► **Theorem 6.2.** *For unary languages, the problem $\mathbf{DFA} \rightarrow \mathbf{NFA}_{\text{syn}}$ is in NP.*

This theorem generalizes the best-known complexity result for unary nfas, which asserts that the problem $\mathbf{DFA} \rightarrow \mathbf{NFA}$ is in NP for *unary cyclic languages* [13], i.e. unary regular languages whose minimal dfa is a cycle. In fact, for any such language L we have shown in [26, Example 5.1] that $\text{nsyn}(L) = \text{ns}(L)$, hence $\mathbf{DFA} \rightarrow \mathbf{NFA}$ coincides with $\mathbf{DFA} \rightarrow \mathbf{NFA}_{\text{syn}}$.

6.3 Group languages

A regular language is called a *group language* if its syntactic monoid forms a group. Several equivalent characterizations of group languages are known; for instance, they are precisely the languages accepted by measure-once quantum finite automata [3]. Concerning their state-minimal (sub)atomic acceptors, we have the following result:

► **Proposition 6.3.** *For any group language L , we have $\text{nsyn}(L) = \text{natm}(L)$.*

Therefore, Theorem 5.2 implies

► **Theorem 6.4.** *For group languages, $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}_{\text{syn}}$ is in NP.*

The complexity of the general $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}_{\text{syn}}$ problem is left as an open problem.

7 Conclusion and Future Work

Approaching from an algebraic and category-theoretic angle we have studied the complexity of computing small (sub)atomic nondeterministic machines. We proved this to be much more tractable than the general case, viz. NP-complete as opposed to PSPACE-complete, provided that one works with a representation of the input language by a pair of dfas or a finite monoid, respectively. There are several interesting directions for future work.

The particular form of our main two NP-complete problems suggests an investigation of their variants $\mathbf{DFA} + \mathbf{DFA}^r \rightarrow \mathbf{NFA}$ and $\mathbf{MON} \rightarrow \mathbf{NFA}$ computing unrestricted nfas. The reductions used in the proof of Theorem 5.2 and 5.7 show both problems to be NP-hard, and we have seen in Theorem 6.1 that they are in NP for nuclear languages. The complexity of the general case is left as an open problem.

The classical algorithm for state minimization of nfas is the Kameda-Weiner method [18], recently given a fresh perspective based on atoms of regular languages [29]. The algorithm involves an enumeration of biclique covers of the dependency relation \mathcal{DR}_L . Since our base equivalence $\mathbf{JSL}_f \simeq \mathbf{Dep}$ reveals a close relationship between biclique covers and semilattice morphisms (e.g. Lemma 3.9), we envision a purely algebraic account of the Kameda-Weiner method. We should also compare our canonical machines to the Universal Automaton [20], a language-theoretic presentation of the Kameda-Weiner algorithm. For example, our morphisms preserve the language whereas the Universal Automaton uses simulations.

Finally, the classes of nuclear and lattice languages – introduced as technical tools for our NP-completeness proofs – deserve to be studied in their own right. For instance, we expect to uncover connections between lattice languages and the characterization of finite simple non-unital semirings which are not rings [31, Theorem 1.7].

References

- 1 Jiří Adámek, Stefan Milius, Robert S. R. Myers, and Henning Urbat. On continuous non-determinism and state minimality. In Bart Jacobs, Alexandra Silva, and Sam Staton, editors, *Proc. 30th Conference on Mathematical Foundations of Programming Science (MFPS'14)*, volume 308 of *Electron. Notes Theor. Comput. Sci.*, pages 3–23. Elsevier, 2014.

- 2 Michael A. Arbib and Ernest G. Manes. Fuzzy machines in a category. *Bulletin of the Australian Mathematical Society*, 13(2):169–210, 1975.
- 3 Alex Brodsky and Nicholas Pippenger. Characterizations of 1-way quantum finite automata. *SIAM J. Comput.*, 31:73–91, 1999.
- 4 Janusz Brzozowski and Hellis Tamm. Theory of átomata. *Theoretical Computer Science*, 539:13–27, 2014.
- 5 Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964.
- 6 Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47:149–158, 1986.
- 7 François Denis, Aurélien Lemay, and Alain Terlutte. Residual finite state automata. In Afonso Ferreira and Horst Reichel, editors, *Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS'01)*, pages 144–157. Springer, 2001.
- 8 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 9 Jaco Geldenhuys, Brink van der Merwe, and Lynette van Zijl. Reducing nondeterministic finite automata with SAT solvers. In Anssi Yli-Jyrä, András Kornai, Jacques Sakarovitch, and Bruce Watson, editors, *Proc. 8th International Workshop on Finite-State Methods and Natural Language Processing (FSMNLP'09)*, pages 81–92. Springer, 2010.
- 10 Hermann Gruber and Markus Holzer. Finding lower bounds for nondeterministic state complexity is hard. In Oscar H. Ibarra and Zhe Dang, editors, *Proc. 10th International Conference on Developments in Language Theory (DLT'06)*, pages 363–374. Springer, 2006.
- 11 Hermann Gruber and Markus Holzer. Computational complexity of NFA minimization for finite and unary languages. In Remco Loos, Szilárd Zsolt Fazekas, and Carlos Martín-Vide, editors, *Proc. 1st International Conference on Language and Automata Theory and Applications (LATA'07)*, pages 261–272. Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona, 2007.
- 12 D.A. Higgs and K.A. Rowe. Nuclearity in the category of complete semilattices. *Journal of Pure and Applied Algebra*, 57(1):67–78, 1989.
- 13 Tao Jiang, Edward McDowell, and B. Ravikumar. The structure and complexity of minimal NFA's over a unary alphabet. *International Journal of Foundations of Computer Science*, 02(02):163–182, 1991.
- 14 Tao Jiang and B. Ravikumar. Minimal NFA problems are hard. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez Artalejo, editors, *Proc. 18th International Colloquium on Automata, Languages, and Programming (ICALP'91)*, pages 629–640. Springer, 1991.
- 15 Tao Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.
- 16 Peter Jipsen. Categories of algebraic contexts equivalent to idempotent semirings and domain semirings. In Wolfram Kahl and Timothy G. Griffin, editors, *Proc. 13th International Conference on Relational and Algebraic Methods in Computer Science (RAMiCS'12)*, pages 195–206. Springer, 2012.
- 17 Peter T. Johnstone. *Stone spaces*. Cambridge University Press, 1982.
- 18 T. Kameda and P. Weiner. On the state minimization of nondeterministic finite automata. *IEEE Transactions on Computers*, C-19(7):617–627, 1970.
- 19 Michel Latteux, Yves Roos, and Alain Terlutte. Minimal NFA and biRFSA languages. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 43(2):221–237, 2009.
- 20 Sylvain Lombardy and Jacques Sakarovitch. The universal automaton. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, pages 457–504, 2008.
- 21 Saunders Mac Lane. *Categories for the working mathematician*. Springer, 2 edition, 1998.
- 22 Andreas Malcher. Minimizing finite automata is computationally hard. *Theor. Comput. Sci.*, 327(3):375–390, 2004.

- 23 George Markowsky. The factorization and representation of lattices. *Transactions of the American Mathematical Society*, 203:185–200, 1975.
- 24 M. Andrew Moshier. A relational category of formal contexts (preprint), 2016.
- 25 Robert S. R. Myers, Jiří Adámek, Stefan Milius, and Henning Urbat. Coalgebraic constructions of canonical nondeterministic automata. *Theor. Comput. Sci.*, 604:81–101, 2015.
- 26 Robert S. R. Myers, Stefan Milius, and Henning Urbat. Nondeterministic syntactic complexity. In Stefan Kiefer and Christine Tasson, editors, *Proc. 24th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'21)*. Springer, 2021. [arXiv:2101.03039](https://arxiv.org/abs/2101.03039).
- 27 Jean-Éric Pin. Mathematical foundations of automata theory. Available at <http://www.liafa.jussieu.fr/~jep/PDF/MPRI/MPRI.pdf>, September 2020.
- 28 K. A. Rowe. Nuclearity. *Canadian Mathematical Bulletin*, 31(2):227–235, 1988.
- 29 Hellis Tamm. New interpretation and generalization of the Kameda-Weiner method. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*, volume 55 of *LIPICs*, pages 116:1–116:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
- 30 Hellis Tamm and Esko Ukkonen. Bideterministic automata and minimal representations of regular languages. *Theoretical Computer Science*, 328(1):135–149, 2004.
- 31 Jens Zumbärgel. Classification of finite congruence-simple semirings with zero. *Journal of Algebra and Its Applications*, 7, March 2007.

Idempotent Turing Machines

Keisuke Nakano   

Research Institute of Electrical Communication, Tohoku University, Sendai, Japan

Abstract

A function f is said to be idempotent if $f(f(x)) = f(x)$ holds whenever $f(x)$ is defined. This paper presents a computation model for idempotent functions, called an idempotent Turing machine. The computation model is necessarily and sufficiently expressive in the sense that not only does it always compute an idempotent function but also every idempotent computable function can be computed by an idempotent Turing machine. Furthermore, a few typical properties of the computation model such as robustness and universality are shown. Our computation model is expected to be a basis of special-purpose (or domain-specific) programming languages in which only but all idempotent computable functions can be defined.

2012 ACM Subject Classification Theory of computation → Turing machines

Keywords and phrases Turing machines, Idempotent functions, Computable functions, Computation model

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.79

Funding This work was partially supported by JSPS KAKENHI Grant Numbers JP21K11744, JP18H04093 and JP17H06099.

Acknowledgements I am grateful to Mirai Ikebuchi for careful proofreading of earlier drafts of this manuscript. I also want to thank anonymous reviewers for their helpful comments.

1 Introduction

A function f whose domain and codomain are equal is said to be *idempotent* if $f(f(x)) = f(x)$ holds whenever $f(x)$ is defined. The idempotence of functions plays an essential role in a wide area of computer science. For example, some program optimization and parallelization do work only when core functions are idempotent; bidirectional transformation is well-behaved only when backward (putback) functions must be idempotent [4, 5, 15]. Moreover, a string sanitizer that removes or escapes potentially dangerous characters to prevent cross-site scripting attacks must be idempotent. Non-idempotent sanitizers are known to make the server vulnerable against double encoding attacks [16]. In such situations, we need to decide if a given function is idempotent or not. However, the idempotence of computable functions is undecidable in general.

To solve the problem, we may design a domain-specific language so that every function defined in the language either is always idempotent as far as it follows the syntax of the language or can be statically verified to be idempotent. Hooimeijer et al. [8] developed the BEK language for describing string sanitizers, which can be checked to perform their appropriate behaviour including idempotence. In this linguistic approach, programmers can define only idempotent functions. However, the restriction may be too strong to exclude idempotent functions that they want to define.

This paper gives a solution to this problem by a computation model, called an *idempotent Turing machine*, which exactly characterizes all idempotent computable functions. More specifically, the computation model is expressive enough for idempotent functions in the sense that every idempotent Turing machine computes an idempotent function and every idempotent computable function can be computed by an idempotent Turing machine. Because of the latter statement, we can claim that a language is sufficiently expressive for idempotent functions if it is capable of simulating any other idempotent Turing machines.



© Keisuke Nakano;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 79; pp. 79:1–79:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The present work follows along the lines of prior work of Axelsen and Glück [2] on *reversible Turing machines*, which are (locally) forward and backward deterministic Turing machines. They have shown that a reversible Turing machine is expressive enough for computing injective functions under *function semantics* under which the meaning of a Turing machine is specified by a function whose input and output correspond to strings on the tape at the initial and final configuration, respectively. We will adopt the function semantics to show the expressiveness of idempotent Turing machines in the present paper. In addition, two more desirable properties of idempotent Turing machines are shown under the function semantics, which have been shown for reversible Turing machines by Axelsen and Glück: the robustness under tape reduction and the existence of a universal machine. The author [14] has also followed along this line to introduce involutory Turing machines as a computation model for involution which is its own inverse.

Our contribution of the present paper is summarized as follows:

- An idempotent Turing machine is proposed as a particular form of a multitape Turing machine. Every idempotent Turing machine computes an idempotent function.
- An idempotent Turing machine is shown to be expressive enough to specify idempotent functions, i.e., every idempotent function is computed by an idempotent Turing machine.
- An idempotent Turing machine is shown to be robust under tape reduction, i.e., every multitape idempotent Turing machine can be simulated by a single-tape Turing machine.
- A universal idempotent Turing machine is shown to exist in terms of an appropriate redefinition of universality [2] i.e., there is an idempotent Turing machine which simulates any other idempotent Turing machine from the description of that machine.

After all of the above are presented, this paper concludes with the related work and a discussion on future work. Some proofs are provided in Appendix A.

2 Preliminaries

An *alphabet* is a finite set of symbols. The set of all strings over an alphabet Σ is denoted by Σ^* . For convenience, we regard a nested tuple of strings as a flattened one, e.g., $((w_1, w_2), w_3)$ and $(w_1, (w_2, w_3))$ may be identified with (w_1, w_2, w_3) for $w_1, w_2, w_3 \in \Sigma^*$.

For a (binary) relation $R \subseteq A \times B$, $a R b$ stands for $(a, b) \in R$. The identity relation $Id_A \subseteq A \times A$ is $\{(a, a) \mid a \in A\}$. The composition of two relations $R \subseteq A \times B$ and $S \subseteq B \times C$, denoted by $S \circ R$, is given as $\{(a, c) \mid \exists b \in B, a R b \wedge b S c\}$. For a relation $R \subseteq A \times B$ over two sets A and B , the *inverse relation* $R^{-1} \subseteq B \times A$ is defined by $\{(b, a) \mid a R b\}$. A relation $R \subseteq A \times A$ is said to be *symmetric* if $R^{-1} = R$. A relation $R \subseteq A \times B$ is said to be *functional* if $a R b_1$ and $a R b_2$ imply $b_1 = b_2$ for any $a \in A$ and $b_1, b_2 \in B$. A functional relation $R \subseteq A \times B$, written by $R : A \rightarrow B$, is simply called a (partial) *function* and $R(a)$ with $a \in A$ stands for $b \in B$ such that $a R b$ if exists. We may write $a \mapsto b$ for an element (a, b) in a functional relation. A function $R : A \rightarrow B$ is said to be *total* if $R(a) \in B$ is defined for any $a \in A$. A function $R : A \rightarrow B$ is said to be *injective* if R^{-1} is functional. For any injective function $R : A \rightarrow B$ it is easy to see that $R^{-1} \circ R \subseteq Id_A$ and $R \circ R^{-1} \subseteq Id_B$ hold. A function $R : A \rightarrow A$ is called *idempotent* if $R \circ R = R$ holds.

3 Turing Machines

The notion of Turing machines is one of the best-known computation model which can implement any computable functions. Many variants of Turing machines have been proposed in the literature in which a single tape or multiple tapes are used, tapes are one-ended or

doubly-infinite, and a head peeks only single cell or multiple adjacent cells. Since they are known to be equi-expressive [17], we basically follow the definition given by Axelsen and Glück [2]. In this section, we define a Turing machine and show its basic properties. We also present a reversible Turing machine, which plays an important role for our main results.

3.1 Syntax and Semantics of Turing machines

A Turing machine manipulates symbols on a doubly-infinite tape of cells according to an internal state and a fixed transition relation.

► **Definition 1** (*k*-tape Turing machine). A *k*-tape Turing machine T is a tuple $(Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ where Q is a finite set of *states*, Σ is a *tape alphabet* not containing the special blank symbol \sqcup , $q_{\text{ini}} \in Q$ is the *initial state*, $q_{\text{fin}} \in Q$ is the *final state*, and $\Delta = \Delta^{\text{sym}} \uplus \Delta^{\leftrightarrow}$ is a ternary relation defining a set of *transition rules* where

$$\begin{aligned} \Delta^{\text{sym}} &\subseteq (Q \setminus \{q_{\text{fin}}\}) \times (\Sigma_{\sqcup} \times \Sigma_{\sqcup})^k \times (Q \setminus \{q_{\text{ini}}\}) \\ \Delta^{\leftrightarrow} &\subseteq (Q \setminus \{q_{\text{fin}}\}) \times \{\leftarrow, \blacklozenge, \rightarrow\}^k \times (Q \setminus \{q_{\text{ini}}\}) \end{aligned}$$

in which Σ_{\sqcup} stands for $\Sigma \uplus \{\sqcup\}$. A *symbol rule* in Δ^{sym} has the form $(q, (s_1 \Rightarrow s'_1, \dots, s_k \Rightarrow s'_k), q')$ with $s_1, \dots, s_k, s'_1, \dots, s'_k \in \Sigma_{\sqcup}$. A *move rule* in Δ^{\leftrightarrow} has the form $(q, (d_1, \dots, d_k), q')$ with $d_1, \dots, d_k \in \{\leftarrow, \blacklozenge, \rightarrow\}$. The second component of a transition rule is called an *action*. In particular, an action in $\{(s, s) \mid s \in \Sigma_{\sqcup}\}^k \cup \{\blacklozenge\}^k$ is called a *null action*.

As presented in [2], transition rules are separated into symbol rules and move rules for our convenience of further discussion, in particular, about the inversion of Turing machines. Although these two kinds of actions are caused by a single rule in ordinary Turing machines [17], the separation of rules does not change the expressiveness of functions. It is easy to simulate a transition rule in an ordinary Turing machine by two transition rules and extra states in the present model.

A configuration of a *k*-tape Turing machine is specified by the current internal state and *k* tapes with their tape head. Each configuration can be characterized by $\langle l, s, r \rangle \in \Sigma_{\sqcup}^{\omega} \times \Sigma_{\sqcup} \times \Sigma_{\sqcup}^{\omega}$ where s is the symbol at its head position and l and r are the left and right tapes of the head. Note that Σ_{\sqcup}^{ω} is a set of infinite strings over Σ_{\sqcup} going infinitely to the right. Accordingly l is “mirrored” where its first symbol is the immediate left one of the head.

► **Definition 2** (Configuration). A *configuration* of a *k*-tape Turing machine $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ is a tuple $(q, \langle l_1, s_1, r_1 \rangle, \dots, \langle l_k, s_k, r_k \rangle)$ where $q \in Q$ is an *internal state*, $l_i, r_i \in \Sigma_{\sqcup}^{\omega}$ for each $i = 1, \dots, k$ are the left and right of the *i*-th tape head, and $s_i \in \Sigma_{\sqcup}$ for each $i = 1, \dots, k$ is the symbol at the *i*-th tape head. The set of all configurations of T is written by \mathcal{C}_T .

► **Definition 3** (Configuration step). Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a *k*-tape Turing machine. Then a single *configuration step* is defined as a relation \vdash_T over \mathcal{C}_T such that

$$(q, \tau_1, \dots, \tau_k) \vdash_T (q', \tau'_1, \dots, \tau'_k)$$

holds for each transition rule $(q, a, q') \in \Delta$ where

- when $a = (s_1 \Rightarrow s'_1, \dots, s_k \Rightarrow s'_k)$, $(\tau_i, \tau'_i) = (\langle l, s_i, r \rangle, \langle l, s'_i, r \rangle)$ holds with some $l, r \in \Sigma_{\sqcup}^{\omega}$ for all $i = 1, \dots, k$;
 - when $a = (d_1, \dots, d_k)$ with $d_i \in \{\leftarrow, \blacklozenge, \rightarrow\}$,
 - $(\tau_i, \tau'_i) = (\langle s'l, s, r \rangle, \langle l, s', sr \rangle)$ holds with some $l, r \in \Sigma_{\sqcup}^{\omega}$ and $s, s' \in \Sigma_{\sqcup}$ if $d_i = \leftarrow$
 - $(\tau_i, \tau'_i) = (\langle l, s, r \rangle, \langle l, s, r \rangle)$ holds with some $l, r \in \Sigma_{\sqcup}^{\omega}$ and $s, s' \in \Sigma_{\sqcup}$ if $d_i = \blacklozenge$
 - $(\tau_i, \tau'_i) = (\langle l, s, s'r \rangle, \langle sl, s', r \rangle)$ holds with some $l, r \in \Sigma_{\sqcup}^{\omega}$ and $s, s' \in \Sigma_{\sqcup}$ if $d_i = \rightarrow$
- for all $i = 1, \dots, k$.

79:4 Idempotent Turing Machines

The subscript T may be omitted if clear from the context.

The semantics of a k -tape Turing machine T is given by a relation over k strings based on \vdash_T^* as follows. In the rest of the paper, a finite string $w \in \Sigma^*$ is used to represent an infinite string $w \sqcup^\omega \in \Sigma_\sqcup^\omega$; thereby, ε denotes \sqcup^ω .

► **Definition 4** (Semantics of Turing machines). Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape Turing machine. The *semantics* of T , denoted by $\llbracket T \rrbracket$, is given by the relation

$$\begin{aligned} \llbracket T \rrbracket = \{ & ((w_1, \dots, w_k), (w'_1, \dots, w'_k)) \in (\Sigma^*)^k \times (\Sigma^*)^k \\ & \mid (q_{\text{ini}}, \langle \varepsilon, \sqcup, w_1 \rangle, \dots, \langle \varepsilon, \sqcup, w_k \rangle) \vdash_T^* (q_{\text{fin}}, \langle \varepsilon, \sqcup, w'_1 \rangle, \dots, \langle \varepsilon, \sqcup, w'_k \rangle) \}. \end{aligned}$$

Recall that we may write $\llbracket T \rrbracket(w_1, \dots, w_k) = (w'_1, \dots, w'_k)$ if $\llbracket T \rrbracket$ is functional.

Following [14], we define the notion of *tidiness* of Turing machines, which is required for further discussion in particular, on the concatenation of Turing machines defined later. Roughly speaking, the tidiness of a Turing machine indicates that the initial configuration is valid if and only if so is the final one. The validity has been called a standard configuration in [1, 2].

► **Definition 5** (Tidiness of Turing machine). A k -tape Turing machine $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ is said to be *tidy* if for any sequence $(q_{\text{ini}}, \langle l_1, s_1, r_1 \rangle, \dots, \langle l_k, s_k, r_k \rangle) \vdash_T^* (q_{\text{fin}}, \langle l'_1, s'_1, r'_1 \rangle, \dots, \langle l'_k, s'_k, r'_k \rangle)$ of computation steps, the following two conditions

- $(l_i, s_i, r_i) \in \{\varepsilon\} \times \{\sqcup\} \times \Sigma^*$ for each $i = 1, \dots, k$
 - $(l'_i, s'_i, r'_i) \in \{\varepsilon\} \times \{\sqcup\} \times \Sigma^*$ for each $i = 1, \dots, k$
- are equivalent.

In the rest of the paper, every k -tape Turing machine is assumed to be tidy. We may call it the *tidiness assumption*.

We shall show two examples of Turing machines whose semantics are both idempotent. As we will see later, due to the form of their transition rules, the second example is an idempotent Turing machine but the first example is not. The main theorem of the present paper claims that any non-idempotent Turing machine has an equivalent idempotent Turing machine whenever its semantics is idempotent, though.

► **Example 6.** The 1-tape Turing machine $T_{\text{ralz}} = (Q, \{0, 1\}, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ where

$$\begin{aligned} Q &= \{q_{\text{ini}}, q_{\text{move}}, q_{\text{ralz}}, q_{\text{back}}, q_{\text{fin}}\} \\ \Delta &= \{(q_{\text{ini}}, \sqcup \rightarrow \sqcup, q_{\text{move}}), (q_{\text{move}}, \rightarrow, q_{\text{ralz}}), (q_{\text{ralz}}, 0 \rightarrow \sqcup, q_{\text{move}}), \\ &\quad (q_{\text{ralz}}, 1 \rightarrow 1, q_{\text{back}}), (q_{\text{ralz}}, \sqcup \rightarrow \sqcup, q_{\text{back}}), (q_{\text{back}}, \leftarrow, q_{\text{fin}})\} \end{aligned}$$

computes the function that removes all leading zeros, i.e., we have $\llbracket T_{\text{ralz}} \rrbracket(0 \dots 0w) = w$ for $w \in \{\varepsilon\} \cup \{1v \mid v \in \{0, 1\}^*\}$.

► **Example 7.** The 2-tape Turing machine $T_{\text{copy}} = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ where

$$\begin{aligned} Q &= \{q_{\text{ini}}, q_{\text{move}}, q_{\text{copy}}, q_{\text{defer}}, q_{\text{trail}}, q_{\text{erase}}, q_{\text{return}}, q_{\text{back}}, q_{\text{check}}, q_{\text{fin}}\} \\ \Delta &= \{(q_{\text{ini}}, (\sqcup \rightarrow \sqcup, \sqcup \rightarrow \sqcup), q_{\text{move}}), (q_{\text{move}}, (\rightarrow, \rightarrow), q_{\text{copy}})\} \cup \\ &\quad \{(q_{\text{copy}}, (s_1 \rightarrow s_2, s_2 \rightarrow s_2), q_{\text{move}}) \mid s_1 \in \Sigma_\sqcup, s_2 \in \Sigma\} \cup \\ &\quad \{(q_{\text{copy}}, (s \rightarrow s, \sqcup \rightarrow \sqcup), q_{\text{trail}}) \mid s \in \Sigma\} \cup \\ &\quad \{(q_{\text{copy}}, (\sqcup \rightarrow \sqcup, \sqcup \rightarrow \sqcup), q_{\text{back}}), (q_{\text{trail}}, (\rightarrow, \rightarrow), q_{\text{defer}})\} \cup \end{aligned}$$

$$\begin{aligned}
& \{(q_{\text{defer}}, (s \Rightarrow s, \sqcup \Rightarrow \sqcup), q_{\text{trail}}) \mid s \in \Sigma\} \cup \\
& \{(q_{\text{defer}}, (\sqcup \Rightarrow \sqcup, \sqcup \Rightarrow \sqcup), q_{\text{return}}), (q_{\text{return}}, (\leftarrow, \leftarrow), q_{\text{erase}})\} \cup \\
& \{(q_{\text{erase}}, (s \Rightarrow \sqcup, \sqcup \Rightarrow \sqcup), q_{\text{return}}) \mid s \in \Sigma\} \cup \{(q_{\text{erase}}, (s \Rightarrow s, s \Rightarrow s), q_{\text{back}}) \mid s \in \Sigma_{\sqcup}\} \cup \\
& \{(q_{\text{check}}, (s \Rightarrow s, s \Rightarrow s), q_{\text{back}}) \mid s \in \Sigma\} \cup \{(q_{\text{back}}, (\leftarrow, \leftarrow), q_{\text{check}}), (q_{\text{check}}, (\sqcup \Rightarrow \sqcup, \sqcup \Rightarrow \sqcup), q_{\text{fin}})\}
\end{aligned}$$

computes the function that copies the second string to the first, i.e., we have $\llbracket T_{\text{copy}} \rrbracket(w_1, w_2) = (w_2, w_2)$ for $w_1, w_2 \in \Sigma^*$. The 2-tape Turing machine T_{copy} can be straightforwardly generalized into $2k$ -tape Turing machine $T_{\text{copy}(k)}$ so that $\llbracket T_{\text{copy}(k)} \rrbracket(w_1, \dots, w_k, v_1, \dots, v_k) = (v_1, \dots, v_k, v_1, \dots, v_k)$ holds for $w_1, \dots, w_k, v_1, \dots, v_k \in \Sigma^*$. In particular, $T_{\text{copy}(1)} = T_{\text{copy}}$.

The second example could be given with fewer states and transition rules by merging q_{defer} with q_{copy} , q_{trail} with q_{move} , q_{erase} with q_{check} , and q_{return} with q_{back} , and removing some redundant rules. However, the smaller alternative is against the condition to be an idempotent Turing machine which will be presented in the next section. The Turing machine T_{copy} of the present form will play an important role in the proof of expressiveness of idempotent Turing machines.

Definition 4 implies that the semantics of a Turing machine returns a tuple that consists of the same number of strings as a given input. However, when the function either only accepts or always returns the empty string on some tapes, we may regard it as a function whose input or output tuple consists of fewer strings following the formalization by Axelsen and Glück [2]. For example, let T be a 3-tape Turing machine over Σ such that $\llbracket T \rrbracket(w_1, w_2, w_3) = (w'_1, w'_2, w'_3)$ implies $w_2 = w_3 = w'_2 = \varepsilon$. Then we may say that T computes a function $f : \Sigma^* \rightarrow \Sigma^* \times \Sigma^*$ defined by $f(w) = (v_1, v_2)$ where $\llbracket T \rrbracket(w, \varepsilon, \varepsilon) = (v_1, \varepsilon, v_2)$ holds. We may simply write $\llbracket T \rrbracket = f$ by ignoring empty input/output strings.

► **Definition 8** (Forward/backward determinism). Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape Turing machine. Then T is *forward deterministic* if, for any distinct pair $(q, a_1, q_1), (q, a_2, q_2) \in \Delta$ of transition rules with the common source state $q \in Q$, their actions a_1 and a_2 have the form of $(s_{1,1} \Rightarrow s'_{1,1}, \dots, s_{1,k} \Rightarrow s'_{1,k})$ and $(s_{2,1} \Rightarrow s'_{2,1}, \dots, s_{2,k} \Rightarrow s'_{2,k})$, respectively, such that $s_{1,i} \neq s_{2,i}$ holds for some $i = 1, \dots, k$. The Turing machine T is *backward deterministic* if, for any distinct pair $(q_1, a_1, q), (q_2, a_2, q) \in \Delta$ of transition rules with the common target state $q \in Q$, their actions a_1 and a_2 have the form of $(s_{1,1} \Rightarrow s'_{1,1}, \dots, s_{1,k} \Rightarrow s'_{1,k})$ and $(s_{2,1} \Rightarrow s'_{2,1}, \dots, s_{2,k} \Rightarrow s'_{2,k})$, respectively, such that $s'_{1,i} \neq s'_{2,i}$ holds for some $i = 1, \dots, k$.

Example 6 and Example 7 are both forward deterministic but not backward deterministic. It is easy to see that every configuration step induced by a forward deterministic Turing machine is functional. In the rest of the paper, we deal with only forward deterministic Turing machines, and hence their semantics are all functional. We may simply say Turing machines even for forward deterministic ones.

Turing machines can be concatenated to synthesize a single one which computes the composition of their semantics.

► **Definition 9** (Concatenation of Turing machines). Let $\{T_i = (Q_i, \Sigma, q_{\text{ini},i}, q_{\text{fin},i}, \Delta_i)\}_{i=1,\dots,n}$ be a family of k -tape Turing machines where Q_1, \dots, Q_n are disjoint without loss of generality. Their *concatenation*, denoted by $T_n \circ \dots \circ T_1$, is a k -tape Turing machine $T = (Q_1 \uplus \dots \uplus Q_n, \Sigma, q_{\text{ini},1}, q_{\text{fin},n}, \Delta)$ where $\Delta = \Delta_1 \uplus \dots \uplus \Delta_n \uplus \{(q_{\text{fin},i-1}, \underbrace{(\blacklozenge, \dots, \blacklozenge)}_k, q_{\text{ini},i}) \mid i = 2, \dots, n\}$.

When Q_1, \dots, Q_n are not disjoint, every state in either should be renamed before the concatenation.

► **Proposition 10** (Semantics of concatenation of Turing machines). *For k -tape reversible Turing machines T_1, \dots, T_n , we have $\llbracket T_n \circ \dots \circ T_1 \rrbracket = \llbracket T_n \rrbracket \circ \dots \circ \llbracket T_1 \rrbracket$.*

Proof. It can be shown straightforwardly with taking notice of the tidiness assumption. ◀

3.2 Reversible Turing Machines

We define a reversible Turing machine, which can be used for the proof of expressiveness of idempotent Turing machines.

► **Definition 11** (Reversible Turing machine). A k -tape Turing machine T is *reversible* if T is forward and backward deterministic.

► **Example 12.** The 2-tape reversible Turing machine $T_{dup} = (\{q_{ini}, q_{move}, q_{copy}, q_{back}, q_{check}, q_{fin}\}, \Sigma, q_{ini}, q_{fin}, \Delta)$ where

$$\begin{aligned} \Delta = & \{(q_{ini}, (\sqcup \rightarrow \sqcup, \sqcup \rightarrow \sqcup), q_{move}), (q_{move}, (\rightarrow, \rightarrow), q_{copy})\} \cup \\ & \{(q_{copy}, (s \rightarrow s, \sqcup \rightarrow s), q_{move}) \mid s \in \Sigma\} \cup \{(q_{copy}, (\sqcup \rightarrow \sqcup, \sqcup \rightarrow \sqcup), q_{back})\} \cup \\ & \{(q_{check}, (s \rightarrow s, s \rightarrow s), q_{back}) \mid s \in \Sigma\} \cup \{(q_{back}, (\leftarrow, \leftarrow), q_{check}), (q_{check}, (\sqcup \rightarrow \sqcup, \sqcup \rightarrow \sqcup), q_{fin})\} \end{aligned}$$

computes the function $\llbracket T_{dup} \rrbracket$, which satisfies $\llbracket T_{dup} \rrbracket(w, \varepsilon) = (w, w)$ for $w \in \Sigma^*$. The 2-tape reversible Turing machine T_{dup} can be straightforwardly generalized into $2k$ -tape reversible Turing machine $T_{dup(k)}$ so that $\llbracket T_{dup(k)} \rrbracket(w_1, \dots, w_k, \underbrace{\varepsilon, \dots, \varepsilon}_k) = (w_1, \dots, w_k, w_1, \dots, w_k)$ holds for $w_1, \dots, w_k \in \Sigma^*$. In particular, we have $T_{dup(1)} = T_{dup}$.

Although the Turing machines T_{dup} and T_{copy} are similar in a sense that they both output pairs of the same string, T_{dup} differs in that the second component of its input is restricted to the empty string. Because of this difference, $\llbracket T_{dup} \rrbracket$ is not idempotent while so is $\llbracket T_{copy} \rrbracket$.

As seen from the definition, a Turing machine obtained by inverting all transition rules of a reversible Turing machine is also reversible. Its semantics is naturally the inverse function of the semantics of the original reversible Turing machine as formally stated below.

► **Definition 13** (Inversion of reversible Turing machines). Let $T = (Q, \Sigma, q_{ini}, q_{fin}, \Delta)$ be a k -tape reversible Turing machine. We define $T^{-1} = (Q, \Sigma, q_{fin}, q_{ini}, \Delta^{-1})$ with $\Delta^{-1} = \{(p, a^{-1}, q) \mid (q, a, p) \in \Delta\}$ where $(a_1, \dots, a_k)^{-1} = (a_1^{-1}, \dots, a_k^{-1})$, $(s \rightarrow s')^{-1} = (s' \rightarrow s)$, $(\leftarrow)^{-1} = (\rightarrow)$, $(\blacklozenge)^{-1} = (\blacklozenge)$, and $(\rightarrow)^{-1} = (\leftarrow)$.

► **Proposition 14** (Bennet [3]). *Let $T = (Q, \Sigma, q_{ini}, q_{fin}, \Delta)$ be a k -tape reversible Turing machine. Then, T^{-1} forms a k -tape reversible Turing machine such that $\llbracket T^{-1} \rrbracket = \llbracket T \rrbracket^{-1}$.*

In reversible Turing machines, by definition, there is no distinct pair of configurations that have the same successive configuration. This implies that the semantics of a reversible Turing machine is always injective.

Axelsen and Glück [2] have shown its converse, i.e., every injective computable function can be defined by a reversible Turing machine. Their proof is constructive so that an equivalent reversible Turing machine can be constructed effectively from a given non-reversible Turing machine whose semantics is injective.

► **Theorem 15** (Expressiveness of reversible Turing machines [2]). *The reversible Turing machines can compute exactly all injective computable functions. That is, given a k -tape Turing machine T such that $\llbracket T \rrbracket$ is injective, there is a k -tape reversible Turing machine T' such that $\llbracket T' \rrbracket = \llbracket T \rrbracket$.*

4 Idempotent Turing Machines

We introduce an idempotent Turing machine and its properties, expressiveness, robustness, and universality.

4.1 Definition and Expressiveness

An idempotent Turing machine is defined by imposing a restriction upon the form of transition rules of a standard Turing machine. Informally, the key of the restriction forces every valid run to include an internal configuration C such that the sequence of configuration steps from C to the final configuration can also become a valid run by concatenating its reversed sequence at the front. Note that in the obtained run, the initial and final tapes contain the same string. This indicates that $\llbracket T \rrbracket(y) = y$ holds for any x and y such that $\llbracket T \rrbracket(x) = y$ for the Turing machine T , which concludes that $\llbracket T \rrbracket$ is idempotent. Formally, idempotent Turing machines are defined as below.

► **Definition 16** (Idempotent Turing machine). Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape forward deterministic Turing machine. Then T is said to be *idempotent* if there exist a set $Q' \subset Q$ of states and a total function $\psi : Q' \rightarrow Q \setminus Q'$ that satisfy the following conditions:

- (I-1) $q_{\text{ini}} \notin Q'$, $q_{\text{fin}} \in Q'$, and $\psi(q_{\text{fin}}) = q_{\text{ini}}$;
- (I-2) there is no transition rule $(q', a, q) \in \Delta$ with $q' \in Q'$ and $q \in Q \setminus Q'$;
- (I-3) there is a transition rule $(\psi(p'), a^{-1}, \psi(q')) \in \Delta$ for each $(q', a, p') \in \Delta$ with $p', q' \in Q'$; and
- (I-4) there is a transition rule $(\psi(q'), a_0, q') \in \Delta$ with a null action a_0 for each $(q, a, q') \in \Delta$ with $q \in Q \setminus Q'$ and $q' \in Q'$.

Each state $q' \in Q'$ is called a *rear state* and the function ψ is called a *rear state map*.

Definition 16 implies that every valid sequence of configuration steps of an idempotent Turing machine can be split into two parts of non-rear and rear states because of the (I-1) and (I-2) conditions. Moreover, with the (I-3) and (I-4) conditions, we can conclude that the semantics of an idempotent Turing machine is idempotent.

► **Theorem 17** (Semantics of idempotent Turing machine). Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape idempotent Turing machine. Then $\llbracket T \rrbracket$ is idempotent.

Proof. For simplicity of the proof, only the case of $k = 1$ is shown. The proof can be easily generalized to the other cases. Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a 1-tape idempotent Turing machine with a set Q' of rear states and a rear state map ψ . It suffices to show that $\llbracket T \rrbracket(v) = v$ holds for any $v = \llbracket T \rrbracket(w)$ with $w \in \Sigma^*$. Suppose that $v = \llbracket T \rrbracket(w)$ for some $v, w \in \Sigma^*$. Because of the definition of idempotent Turing machines, there must exist a valid sequence of configuration steps of the form

$$(q_{\text{ini}}, \langle \varepsilon, \sqcup, w \rangle) \vdash (q_1, \langle l_1, s_1, r_1 \rangle) \vdash \dots \vdash (q_m, \langle l_m, s_m, r_m \rangle) \vdash \\ (q'_1, \langle l'_1, s'_1, r'_1 \rangle) \vdash \dots \vdash (q'_n, \langle l'_n, s'_n, r'_n \rangle) \vdash (q_{\text{fin}}, \langle \varepsilon, \sqcup, v \rangle)$$

with $q_1, \dots, q_m \in Q \setminus Q'$ and $q'_1, \dots, q'_n \in Q'$ from the (I-1) and (I-2) conditions. Because of the (I-4) condition, there is a transition rule $(\psi(q'_1), a_0, q'_1) \in \Delta$ with a null action a_0 . Then we have a valid sequence

$$(q_{\text{ini}}, \langle \varepsilon, \sqcup, v \rangle) \vdash (\psi(q'_n), \langle l'_n, s'_n, r'_n \rangle) \vdash \dots \vdash (\psi(q'_1), \langle l'_1, s'_1, r'_1 \rangle) \vdash \\ (q'_1, \langle l'_1, s'_1, r'_1 \rangle) \vdash \dots \vdash (q'_n, \langle l'_n, s'_n, r'_n \rangle) \vdash (q_{\text{fin}}, \langle \varepsilon, \sqcup, v \rangle)$$

due to the (I-1) and (I-3) conditions, which demonstrates $\llbracket T \rrbracket(v) = v$. ◀

79:8 Idempotent Turing Machines

The Turing machine T_{copy} introduced in Example 7 can be shown to be idempotent accompanied by a set Q' of rear states and a rear state map ψ specified by

$$Q' = \{q_{fin}, q_{check}, q_{back}\} \quad \psi = \{q_{fin} \mapsto q_{ini}, q_{check} \mapsto q_{move}, q_{back} \mapsto q_{copy}\}.$$

The conditions **(I-1)** and **(I-2)** obviously hold. The condition **(I-3)** can be confirmed by the correspondence of the pairs of transition rules,

$$\begin{array}{ll} (q_{check}, (\sqcup \rightarrow \sqcup, \sqcup \rightarrow \sqcup), q_{fin}) & (q_{ini}, (\sqcup \rightarrow \sqcup, \sqcup \rightarrow \sqcup), q_{move}) \\ (q_{back}, (\leftarrow, \leftarrow), q_{check}) & (q_{move}, (\rightarrow, \rightarrow), q_{copy}) \\ (q_{check}, (s \rightarrow s, s \rightarrow s), q_{back}) & (q_{copy}, (s \rightarrow s, s \rightarrow s), q_{move}) \end{array}$$

with $s \in \Sigma$. The condition **(I-4)** holds because of the transition rule $(q_{copy}, (\sqcup \rightarrow \sqcup, \sqcup \rightarrow \sqcup), q_{back})$. The general Turing machine $T_{copy(k)}$ can be checked to be idempotent as well.

In contrast, the 1-tape Turing machine T_{ralz} introduced in Example 6 and the smaller alternative of T_{copy} mentioned after Example 7 are not idempotent even though their semantics is idempotent. As for T_{ralz} , we can check it as follows. In order to satisfy the conditions **(I-1)** and **(I-3)**, the q_{back} state cannot be a rear state. Then it is impossible to satisfy the condition **(I-4)**. In general we can decide whether a given Turing machine is idempotent.

► **Proposition 18** (Decidability of idempotence of Turing machines). *Let T be a k -tape Turing machine. It is decidable whether T is idempotent.*

Proof. The proof depends on the finiteness of the set of states and accordingly the finiteness of the choice of the set of rear states and the rear state map, which is required to be idempotent. ◀

The proof above indicates just the existence an extremely naive procedure for the decision problem. The complexity of the decision procedure is beyond double exponential to the number of states. We leave for future work a more efficient algorithm.

Note that the decision problem is only to decide if a Turing machine is idempotent but not to decide if the Turing machine computes an idempotent function. The latter problem is obviously undecidable. However, we will prove that an equivalent idempotent Turing machine can be constructed whenever the Turing machine computes an idempotent function.

Idempotent functions are closed under *conjugation* with injective functions, i.e., for any idempotent function f and injective function g , the conjugate $g^{-1} \circ f \circ g$ is idempotent. The following lemma shows that the idempotent Turing machines have a similar property which will be used to prove the main theorem. The proof of this lemma is given in Appendix A.

► **Lemma 19** (Closed under conjugation). *Let T be a k -tape idempotent Turing machine. For any k -tape reversible Turing machine T_r , the k -tape reversible Turing machine $T_r^{-1} \circ T \circ T_r$ is idempotent.*

Now we are ready to prove expressiveness of the idempotent Turing machines which is one of the main theorems in the present paper.

► **Theorem 20** (Expressiveness of idempotent Turing machines). *The idempotent Turing machines can compute any idempotent computable function. More specifically, given a k -tape Turing machine T such that $\llbracket T \rrbracket$ is idempotent, there is a $2k$ -tape idempotent Turing machine T' such that $\llbracket T' \rrbracket = \llbracket T \rrbracket$.*

Proof sketch. Let T be a k -tape Turing machine such that $\llbracket T \rrbracket : \Sigma^k \rightarrow \Sigma^k$ is idempotent. Consider a (partial) function $f : \Sigma^{2k} \rightarrow \Sigma^{2k}$ such that $f(w_1, \dots, w_k, w_{k+1}, \dots, w_{2k}) = (w_1, \dots, w_k, \llbracket T \rrbracket(w_1, \dots, w_k))$ holds only if $\llbracket T \rrbracket(w_1, \dots, w_k)$ is defined and $w_{k+1} = \dots = w_{2k} = \varepsilon$ holds. Since the function f is injective and computable, we can construct a $2k$ -tape reversible Turing machine T_f such that $\llbracket T_f \rrbracket = f$ by Theorem 15. Then, define a $2k$ -tape Turing machine $T' = T_f^{-1} \circ T_{\text{copy}(k)} \circ T_f$ where $T_{\text{copy}(k)}$ is a $2k$ -tape idempotent Turing machine introduced in Example 7. The Turing machine T' is idempotent due to Lemma 19 and a simple calculation can verify $\llbracket T' \rrbracket = \llbracket T \rrbracket$ as shown in Appendix A. ◀

4.2 Robustness under Tape Reduction

Single-tape Turing machines are as expressive as multitape Turing machines [17]. This property is known as one of the robustness of Turing machines. We will see the property for idempotent Turing machines, i.e., every multitape idempotent Turing machine has an equivalent single-tape idempotent Turing machine. To this end, we simulate a k -tuple of strings with a single string by an encoding function $\text{enc} : (\Sigma^*)^k \rightarrow (\Sigma \uplus \{\$, \})^*$ using a special symbol $\$$ not in Σ . The encoding function is defined as

$$\text{enc}(s_{1,1}s_{1,2}\dots s_{1,n}, \dots, s_{k,1}s_{k,2}\dots s_{k,n}) = s_{1,1}\dots s_{k,1}s_{1,2}\dots s_{k,2}s_{1,n}\dots s_{k,n}$$

with the maximum length n of the input strings where the symbol $\$$ is filled at the end of the shorter strings as necessary, e.g., $\text{enc}(\text{ab}, \text{cdef}, \text{ghi}) = \text{acgbdh\$ei\$f\$}$. The encoding function is injective and computable where k is fixed. We will write $\Sigma_{\$}$ for $\Sigma \uplus \{\$, \}$ and enc may be used even for encoded strings, i.e., $\text{enc} : (\Sigma_{\$}^*)^k \rightarrow \Sigma_{\* .

We first show how to construct a 2-tape idempotent Turing machine equivalent to a given multitape idempotent Turing machine. This is easily shown by the expressiveness of idempotent Turing machines.

► **Theorem 21** (Reduction to 2-tape idempotent Turing machine). *Let T be a k -tape idempotent Turing machine. Then there exists a 2-tape idempotent Turing machine T' that simulates T , that is, $\llbracket T \rrbracket(w_1, \dots, w_k) = (v_1, \dots, v_k)$ if and only if $\llbracket T' \rrbracket(\text{enc}(w_1, \dots, w_k), \varepsilon) = (\text{enc}(v_1, \dots, v_k), \varepsilon)$.*

Proof. Let T be a k -tape idempotent Turing machine. Consider a function $f : \Sigma_{\$}^* \rightarrow \Sigma_{\* satisfying $f(\text{enc}(w_1, \dots, w_k)) = \text{enc}(\llbracket T \rrbracket(w_1, \dots, w_k))$ for any $w_1, \dots, w_k \in \Sigma^*$ only if $\llbracket T \rrbracket(w_1, \dots, w_k)$ is defined. Since f is computable, we have a 1-tape Turing machine that computes f . Note that f is idempotent because of the idempotence of $\llbracket T \rrbracket$. Therefore, there exists a 2-tape idempotent Turing machine that computes f by Theorem 20 with $k = 1$. ◀

The theorem above is not satisfactory because it still requires at least two tapes to simulate arbitrary multitape idempotent Turing machines. We need a further idea to show the robustness under the tape reduction down to a single tape. The idea is similar to that of the proof for expressiveness. We employ a 1-tape idempotent Turing machine T_{blur} , which behaves like T_{copy} as shown by the following lemma. Its proof is given in Appendix A.

► **Lemma 22.** *There exists a 1-tape idempotent Turing machine T_{blur} computing the idempotent function $f_{\text{blur}} : \Sigma_{\$}^* \rightarrow \Sigma_{\* , which satisfies $f_{\text{blur}}(\varepsilon) = \varepsilon$ and $f_{\text{blur}}(s_1s_2w) = s_1s_1f_{\text{blur}}(w)$ for any $s_1, s_2 \in \Sigma_{\$}$ and $w \in \Sigma_{\* .*

Now we are ready to prove the robustness under tape reduction for idempotent Turing machines. In the proof of the robustness theorem, the idempotent Turing machine T_{blur} plays a similar role to $T_{\text{copy}(k)}$ in the proof of Theorem 20.

► **Theorem 23** (Robustness of idempotent Turing machines). *Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape idempotent Turing machine. Then there exists a 1-tape idempotent Turing machine T' such that $\llbracket T \rrbracket(w_1, \dots, w_k) = (v_1, \dots, v_k)$ if and only if $\llbracket T' \rrbracket(\text{enc}(w_1, \dots, w_k)) = \text{enc}(v_1, \dots, v_k)$.*

Proof sketch. Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape idempotent Turing machine. Consider a function $f : \Sigma_{\#}^* \rightarrow \Sigma_{\#}^*$ satisfying $f(\text{enc}(w_1, \dots, w_k)) = \text{enc}(\llbracket T \rrbracket(w_1, \dots, w_k))$, $\text{enc}(w_1, \dots, w_k)$ for any $w_1, \dots, w_n \in \Sigma^*$ only if $\llbracket T \rrbracket(w_1, \dots, w_k)$ is defined. Since the function f is injective and computable, we can construct a 1-tape Turing machine T_f such that $\llbracket T_f \rrbracket = f$ by Theorem 15. Then, define a 1-tape Turing machine $T' = T_f^{-1} \circ T_{\text{blur}} \circ T_f$ where T_{blur} is an idempotent Turing machine given in Lemma 22. The Turing machine T' is idempotent because of Lemma 19 and a simple calculation can verify $\llbracket T' \rrbracket \circ \text{enc} = \text{enc} \circ \llbracket T \rrbracket$ as shown in Appendix A. ◀

4.3 Universality

A standard Turing machine is called *universal* if it is capable of simulating an arbitrary Turing machine on arbitrary input. A universal Turing machine takes a pair of strings: one is the description of the given Turing machine T , which is typically provided as the Gödel number $\ulcorner T \urcorner$ as a string; another is an input w of T . Then it is expected to return the output string $\llbracket T \rrbracket(w)$. In essence, a universal Turing machine U must satisfy $\llbracket U \rrbracket(\ulcorner T \urcorner, x) = \llbracket T \rrbracket(w)$ for any Turing machine T and its input string w .

With regard to idempotent Turing machines, there is no universal machines in the sense above, i.e., no idempotent Turing machine simulate arbitrary idempotent Turing machines. Since the domain and codomain of idempotent functions must be equal, the universal machine cannot be idempotent. Therefore, we relax the definition of universality as Axelsen and Glück have done to introduce the universality of reversible Turing machines [2] where the universal machine returns not only the expected output string but also the given Turing machine itself. Under this relaxed definition, the universal machine computes an idempotent function as far as the given Turing machine is idempotent.

► **Definition 24** (Universality). *A k -tape idempotent Turing machine U is said to be *IdTM-universal* if $\llbracket U \rrbracket(\ulcorner T \urcorner, w) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w))$ holds for any idempotent Turing machine T and its input string w .*

In the present paper, a universal model of ordinary Turing machines is called a *classically universal Turing machine* to distinguish from our IdTM-universal machines.

It is not difficult to show the existence of an IdTM-universal machine because of the expressiveness of idempotent Turing machines.

► **Theorem 25.** *There exists an IdTM-universal idempotent Turing machine.*

Proof. Let f be a function satisfying $f(\ulcorner T \urcorner, w) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w))$ for any idempotent Turing machine T and its input string w . Note that f is idempotent because $f(f(\ulcorner T \urcorner, w)) = f(\ulcorner T \urcorner, \llbracket T \rrbracket(w)) = (\ulcorner T \urcorner, \llbracket T \rrbracket(\llbracket T \rrbracket(w))) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w))$ where the last equality comes from the idempotence of $\llbracket T \rrbracket$. By Theorem 20, we obtain an idempotent Turing machine U that computes f . ◀

The theorem above just shows the existence of an universal machine, which is considered impractical as noticed in the prior work [1, 2, 14] because its proof relies on Theorem 15 (via Theorem 20), which is the expressive theorem of reversible Turing machine. As mentioned in [2], the proof of Theorem 15 is based on very inefficient generate-and-test method by

McCarthy [12]. To avoid this problem, we shall show the construction of an universal Turing machine without Theorem 15 where we construct a universal idempotent Turing machine from a classically universal Turing machine. Following the existing work [1, 2, 14], we employ Bennett's trick with Landauer's trace embedding to construct a special type of reversible Turing machines that computes the function $\lambda x.(x, f(x))$ for an arbitrary computable function f . We present a multitape version of the theorem as below. Its proof is provided in Appendix A.

► **Proposition 26** (Bennett's trick [3]). *Let T be a k -tape Turing machine. There exists a $(2k + 1)$ -tape reversible Turing machine $\text{Ben}_k(T)$ such that $\llbracket \text{Ben}_k(T) \rrbracket(w_1, \dots, w_k, \underbrace{\varepsilon, \dots, \varepsilon}_{k+1}) = (w_1, \dots, w_k, \llbracket T \rrbracket(w_1, \dots, w_k), \varepsilon)$ for any input w_1, \dots, w_k of T .*

With the Bennett's trick, we can construct a universal idempotent Turing machine from a classically universal Turing machine.

► **Theorem 27** (IdTM-universal Turing machine constructed with Bennett's trick). *Let U be a (2-tape) classically universal Turing machine, i.e., $\llbracket U \rrbracket(\ulcorner T \urcorner, w) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w))$ for any Turing machine T and its input w . Then a 5-tape idempotent Turing machine $U' = \text{Ben}_2(U)^{-1} \circ T'_{\text{copy}(2)} \circ \text{Ben}_2(U)$ is IdTM-universal where $T'_{\text{copy}(2)}$ is obtained by adding one extra tape as the fifth tape to $T_{\text{copy}(2)}$, i.e., $\llbracket T'_{\text{copy}(2)} \rrbracket(w_1, w_2, w_3, w_4, w_5) = (w_3, w_4, w_3, w_4, w_5)$.*

Proof sketch. Let U be a 2-tape classically universal Turing machine such that $\llbracket U \rrbracket(\ulcorner T \urcorner, w) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w))$ for any Turing machine T and its input w . Note that the Turing machine $U' = \text{Ben}_2(U)^{-1} \circ T'_{\text{copy}(2)} \circ \text{Ben}_2(U)$ is idempotent due to the idempotence of $T'_{\text{copy}(2)}$ and Lemma 19. In addition, we can show the 5-tape idempotent Turing machine U' is IdTM-universal, that is, $\llbracket U' \rrbracket(\ulcorner T \urcorner, w, \varepsilon, \varepsilon, \varepsilon) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w), \varepsilon, \varepsilon, \varepsilon)$ holds for any Turing machine T and its input w as shown in Appendix A. ◀

The author [14] has constructed a universal machine for involutory Turing machines using Bennett's trick in a way similar to Theorem 27. The difference is that he uses the Turing machine permuting some of tapes for the center one of the composition instead of $T'_{\text{copy}(2)}$.

5 Related work

This work proposes idempotent Turing machines, which can compute exactly all idempotent computable functions. The present work has followed along the lines of prior work on special Turing machines for particular classes of computable functions [1, 2, 14] under function semantics, in which the meaning of a Turing machine is specified by a function whose input and output correspond to strings on the tape at the initial and final configuration, respectively.

Axelsen and Glück [1, 2] have investigated several properties of reversible Turing machines under function semantics. Even though the notion of reversible Turing machines had been already introduced and studied before the work [3, 11], Axelsen and Glück gave much clearer semantics to reversible Turing machines to observe what they compute. They showed that reversible Turing machines can compute exactly all injective computable functions. They also proved the robustness under tape and symbol reduction: 1-tape 3-symbol reversible Turing machines can simulate arbitrary multitape reversible Turing machines. Furthermore, they showed the existence of universal reversible Turing machines under an appropriate redefinition of universality and gave an efficient construction of a universal machine. The present work has followed their approach and utilized their results even though idempotent functions are not necessarily injective.

79:12 Idempotent Turing Machines

The author [14] introduced involutory Turing machines and showed that they can compute exactly all involutory computable functions, which are own inverse, i.e., a function f such that $f(f(x)) = x$ holds whenever $f(x)$ is defined. He naturally followed Axelsen and Glück's work since every involutory function is injective. He defined an involutory Turing machine as a Turing machine whose transition rules are related each other under an involutory map over states. The idea is found in discrete time-symmetric systems [6,9]. In the present work, we refer to the idea to define an idempotent Turing machine with the rear state map.

Besides Turing machines, there are many other model of computable functions, e.g., untyped lambda calculus, combinatory logic, and term/string rewriting systems. We could consider a restricted model of them for idempotent functions. We have selected Turing machines because there is a well-studied subclass of the model, namely reversible Turing machine, for injective functions that can be invertible. A candidate of other such models would be Mu et al's injective language `Inv` [13] in which every function is defined by a few primitives including the fixed-point operator. They have shown that the `Inv` language is expressive enough to simulate reversible Turing machines. However, it is not simple to utilize the injective language for a computational model of idempotent functions. We need to add more primitives to describe non-injective functions and impose some syntactic restrictions to define only idempotent functions. It would be interesting if such a programming language can be defined, though.

6 Conclusion

We have introduced a computation model for idempotent functions, called an idempotent Turing machine. This model is necessarily and sufficiently expressive: every idempotent Turing machine computes an idempotent function and every idempotent function can be computed by an idempotent Turing machine. The class of Turing machines has been shown to be robust under tape reduction. We have also shown the existence of an universal idempotent Turing machine and its construction.

Our computation model is expected to be a basis of special-purpose (or domain-specific) programming languages in which only but all idempotent computable functions can be defined. A computation model is said to be *Turing-complete* if it can simulate any Turing machine. The notion of Turing-completeness is often used to show the expressiveness of not only a computation model but also a programming language or a set of machine instructions. For reversible computation, the notion of *r-Turing completeness* has been proposed [1,2,19]. There have been reversible programming languages, e.g., Janus with dynamic storage [18], reversible flowchart language [19], and R-WHILE [7], that have been shown to be *r-Turing complete*. Similarly, the notion of *idempotent-Turing-completeness* can be defined and applied to a special-purpose (or domain-specific) programming languages in which only but all idempotent computable functions can be defined. Such a special programming language will be required for string sanitizers, automated program optimizers, and bidirectional transformation.

In addition, it is interesting to consider computational models that exactly cover all computable functions with some constraints in the general case. Axelsen and Glück [1] have shown that reversible Turing machines can exactly cover injective computable functions. The author [14] has shown a computational model that exactly covers involutory computable functions, and he shows such a result for idempotent functions in the present paper. It is a natural question to ask what kind of semantic constraints on computable functions corresponds to syntactically constrained Turing machines in general, which is left for future work.

References

- 1 Holger Bock Axelsen and Robert Glück. What Do Reversible Programs Compute? In *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, pages 42–56, 2011. doi:10.1007/978-3-642-19805-2_4.
- 2 Holger Bock Axelsen and Robert Glück. On reversible Turing machines and their function universality. *Acta Inf.*, 53(5):509–543, 2016. doi:10.1007/s00236-015-0253-y.
- 3 Charles H Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532, 1973. doi:10.1147/rd.176.0525.
- 4 Sebastian Fischer, Zhenjiang Hu, and Hugo Pacheco. The essence of bidirectional programming. *Sci. China Inf. Sci.*, 58(5):1–21, 2015. doi:10.1007/s11432-015-5316-8.
- 5 J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Trans. Program. Lang. Syst.*, 29(3):17, 2007. doi:10.1145/1232420.1232424.
- 6 Anahí Gajardo, Jarkko Kari, and Andrés Moreira. On time-symmetry in cellular automata. *J. Comput. Syst. Sci.*, 78(4):1115–1126, 2012. doi:10.1016/j.jcss.2012.01.006.
- 7 Robert Glück and Tetsuo Yokoyama. A Linear-Time Self-Interpreter of a Reversible Imperative Language. *Computer Software*, 33(3):3_108–3_128, 2016. doi:10.11309/jssst.33.3_108.
- 8 Pieter Hooimeijer, Benjamin Livshits, David Molnar, Prateek Saxena, and Margus Veanes. Fast and Precise Sanitizer Analysis with BEK. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. USENIX Association, 2011.
- 9 Martin Kutrib and Thomas Worsch. Time-Symmetric Machines. In *Reversible Computation - 5th International Conference, RC 2013, Victoria, BC, Canada, July 4-5, 2013. Proceedings*, pages 168–181, 2013. doi:10.1007/978-3-642-38986-3_14.
- 10 R. Landauer. Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, 5(3):183–191, July 1961. doi:10.1147/rd.53.0183.
- 11 Yves Lecerf. Machines de Turing réversibles. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 257:2597–2600, 1963.
- 12 John McCarthy. The Inversion of Functions Defined by Turing Machines. In *Automata Studies. (AM-34)*, pages 177–182. Princeton University Press, 1956. doi:10.1515/9781400882618-009.
- 13 Shin-Cheng Mu, Zhenjiang Hu, and Masato Takeichi. An Injective Language for Reversible Computation. In Dexter Kozen and Carron Shankland, editors, *Mathematics of Program Construction, 7th International Conference, MPC 2004, Stirling, Scotland, UK, July 12-14, 2004, Proceedings*, volume 3125 of *Lecture Notes in Computer Science*, pages 289–313. Springer, 2004. doi:10.1007/978-3-540-27764-4_16.
- 14 Keisuke Nakano. Involutionary Turing Machines. In Ivan Lanese and Mariusz Rawski, editors, *Reversible Computation - 12th International Conference, RC 2020, Oslo, Norway, July 9-10, 2020, Proceedings*, volume 12227 of *Lecture Notes in Computer Science*, pages 54–70. Springer, 2020. doi:10.1007/978-3-030-52482-1_3.
- 15 Keisuke Nakano. A Tangled Web of 12 Lens Laws. In Shigeru Yamashita and Tetsuo Yokoyama, editors, *Reversible Computation - 13th International Conference, RC 2021, Virtual Event, July 7-8, 2021, Proceedings*, volume 12805 of *Lecture Notes in Computer Science*, pages 185–203. Springer, 2021. doi:10.1007/978-3-030-79837-6_11.
- 16 OWASP. Double Encoding. https://owasp.org/www-community/Double_Encoding. [Online; 21-January-2021].
- 17 Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, Boston, MA, 1997.
- 18 Tetsuo Yokoyama, Holger Bock Axelsen, and Robert Glück. Principles of a reversible programming language. In *Proceedings of the 5th Conference on Computing Frontiers, 2008, Ischia, Italy, May 5-7, 2008*, pages 43–54, 2008. doi:10.1145/1366230.1366239.

- 19 Tetsuo Yokoyama, Holger Bock Axelsen, and Robert Glück. Reversible Flowchart Languages and the Structured Reversible Program Theorem. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 258–270. Springer, 2008. doi:10.1007/978-3-540-70583-3_22.

A Proofs of Lemmas, Theorems and Proposition

This Appendix provides proofs that are omitted or condensed in the main text.

► **Lemma 19** (Closed under conjugation). *Let T be a k -tape idempotent Turing machine. For any k -tape reversible Turing machine T_r , the k -tape reversible Turing machine $T_r^{-1} \circ T \circ T_r$ is idempotent.*

Proof. Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape idempotent Turing machine with a set $Q' \subset Q$ of rear states and a rear state map ψ . Let $T_r = (Q_r, \Sigma_r, q_{\text{ini},r}, q_{\text{fin},r}, \Delta_r)$ be a k -tape reversible Turing machine where Q and Q_r are disjoint without loss of generality. In order to concatenate T_r , T , and T_r^{-1} following Definition 9, we rename every state $q \in Q_r$ of T_r^{-1} with \widehat{q} . Let \widehat{Q}_r be a set of states of T_r^{-1} , i.e., $\widehat{Q}_r = \{\widehat{q} \mid q \in Q_r\}$ and $\Delta_r^{-1} = \{(\widehat{p}, a^{-1}, \widehat{q}) \mid (q, a, p) \in \Delta_r\}$. Then we can define $T_c = T_r^{-1} \circ T \circ T_r$, which is to be shown idempotent, as $T_c = (Q_c, \Sigma, q_{\text{ini},r}, \widehat{q_{\text{ini},r}}, \Delta_c)$ where

$$\begin{aligned} Q_c &= Q_r \uplus Q \uplus \widehat{Q}_r \\ \Delta_c &= \Delta_r \uplus \Delta \uplus \{(\widehat{p}, a^{-1}, \widehat{q}) \mid (q, a, p) \in \Delta_r\} \uplus \{(q_{\text{fin},r}, \underbrace{(\diamond, \dots, \diamond)}_k), q_{\text{ini}}\}, \{(q_{\text{fin}}, \underbrace{(\diamond, \dots, \diamond)}_k), \widehat{q_{\text{fin},r}}\}. \end{aligned}$$

Let Q'_c be a subset of Q_c defined by $Q' \uplus \widehat{Q}_r$. We define a function $\psi_c : Q'_c \rightarrow Q_c \setminus Q'_c$ as

$$\psi_c(q) = \begin{cases} \psi(q) & (q \in Q') \\ q_r & (q = \widehat{q}_r \in \widehat{Q}_r) \end{cases}$$

where the codomain of ψ_c is $(Q \setminus Q') \uplus Q_r$ that is equal to $Q_c \setminus Q'_c$.

We shall check the four conditions of Q'_c and ψ_c for T_c to be idempotent.

- The **(I-1)** condition holds since $q_{\text{ini},r} \in Q_r \subset Q_c \setminus Q'_c$, $\widehat{q_{\text{ini},r}} \in \widehat{Q}_r \subset Q'_c$, and $\psi_c(\widehat{q_{\text{ini},r}}) = q_{\text{ini},r}$.
- The **(I-2)** condition holds because of the definition of concatenation and the **(I-2)** condition for T .
- Concerning the **(I-3)** condition, suppose that we have $(q', a, p') \in \Delta_c$ with $p', q' \in Q'_c = Q' \uplus \widehat{Q}_r$. Since it is impossible to have $p' \in Q' \subset Q$ and $q' \in \widehat{Q}_r$ due to the construction of Δ_c , we divide it into three cases:
 - When $p', q' \in Q'$, we have $(\psi_c(p'), a^{-1}, \psi_c(q')) \in \Delta_c$ because of the **(I-3)** condition of T with $\psi_c(p') = \psi(p')$, $\psi_c(q') = \psi(q')$, and $\Delta \subset \Delta_c$;
 - When $p' \in \widehat{Q}_r$ and $q' \in Q'$, we must have $p' = \widehat{q_{\text{fin},r}}$ and $q' = q_{\text{fin}}$ according to the construction of Δ_c . Hence, we have $(\psi_c(p'), a^{-1}, \psi_c(q')) \in \Delta_c$ because of $\psi_c(p') = q_{\text{fin},r}$, $\psi_c(q') = \psi(q_{\text{fin}}) = q_{\text{ini}}$, and the construction of Δ_c ; and
 - When $p', q' \in \widehat{Q}_r$, there exist $p_r, q_r \in Q_r$ such that $p' = \widehat{p}_r$ and $q' = \widehat{q}_r$. Hence, we have $(\psi_c(p'), a^{-1}, \psi_c(q')) \in \Delta_c$ because of $\psi_c(p') = p_r$, $\psi_c(q') = q_r$, and the construction of Δ_c with Δ_r .

Thus, the condition **(I-3)** holds.

- Concerning **(I-4)** condition, suppose that we have $(q, a, q') \in \Delta_c$ with $q \in (Q_c \setminus Q') = (Q \setminus Q' \uplus Q_r)$ and $q' \in Q'_c = Q' \uplus \widehat{Q}_r$. From the construction of Δ_c , we must have $q \in Q \setminus Q'$ and $q' \in Q'$. Thus, we obtain $q = \psi(q') = \psi_c(q')$ and $a = a^{-1}$ because of the condition **(I-4)** of T .

Therefore we conclude that T_c is idempotent with the set Q_c of rear states and the rear state map ψ_c . ◀

► **Theorem 20** (Expressiveness of idempotent Turing machines). *The idempotent Turing machines can compute any idempotent computable function. More specifically, given a k -tape Turing machine T such that $\llbracket T \rrbracket$ is idempotent, there is a $2k$ -tape idempotent Turing machine T' such that $\llbracket T' \rrbracket = \llbracket T \rrbracket$.*

Proof. Let T be a k -tape Turing machine such that $\llbracket T \rrbracket : \Sigma^k \rightarrow \Sigma^k$ is idempotent. Consider a (partial) function $f : \Sigma^{2k} \rightarrow \Sigma^{2k}$ such that $f(w_1, \dots, w_k, w_{k+1}, \dots, w_{2k}) = (w_1, \dots, w_k, \llbracket T \rrbracket(w_1, \dots, w_k))$ holds only if $\llbracket T \rrbracket(w_1, \dots, w_k)$ is defined and $w_{k+1} = \dots = w_{2k} = \varepsilon$ holds. Since the function f is injective and computable, we can construct a $2k$ -tape reversible Turing machine T_f such that $\llbracket T_f \rrbracket = f$ by Theorem 15.

Let us define a $2k$ -tape Turing machine $T' = T_f^{-1} \circ T_{\text{copy}(k)} \circ T_f$ where $T_{\text{copy}(k)}$ is a $2k$ -tape idempotent Turing machine introduced in Example 7. Since T' is idempotent because of Lemma 19, it suffices to show $\llbracket T' \rrbracket = \llbracket T \rrbracket$, i.e., $\llbracket T' \rrbracket(w_1, \dots, w_k, \underbrace{\varepsilon, \dots, \varepsilon}_k) = (\llbracket T \rrbracket(w_1, \dots, w_k), \underbrace{\varepsilon, \dots, \varepsilon}_k)$. This can be shown by

$$\begin{aligned}
\llbracket T' \rrbracket(w_1, \dots, w_k, \varepsilon, \dots, \varepsilon) &= \{ \text{by the definition of } T' \text{ and Proposition 10} \} \\
&\quad \llbracket T_f^{-1} \rrbracket(\llbracket T_{\text{copy}(k)} \rrbracket(\llbracket T_f \rrbracket(w_1, \dots, w_k, \varepsilon, \dots, \varepsilon))) \\
&= \{ \text{by the definition of } T_f \text{ and Proposition 14} \} \\
&\quad \llbracket T_f \rrbracket^{-1}(\llbracket T_{\text{copy}(k)} \rrbracket(w_1, \dots, w_k, \llbracket T \rrbracket(w_1, \dots, w_k))) \\
&= \{ \text{by the definition of } T_{\text{copy}(k)} \} \\
&\quad \llbracket T_f \rrbracket^{-1}(\llbracket T \rrbracket(w_1, \dots, w_k), \llbracket T \rrbracket(w_1, \dots, w_k)) \\
&= \{ \text{by the idempotence of } \llbracket T \rrbracket \} \\
&\quad \llbracket T_f \rrbracket^{-1}(\llbracket T \rrbracket(w_1, \dots, w_k), \llbracket T \rrbracket(\llbracket T \rrbracket(w_1, \dots, w_k))) \\
&= \{ \text{by the semantics of } T_f \text{ and its injectivity} \} \\
&\quad (\llbracket T \rrbracket(w_1, \dots, w_k), \varepsilon, \dots, \varepsilon). \quad \blacktriangleleft
\end{aligned}$$

► **Lemma 22.** *There exists a 1-tape idempotent Turing machine T_{blur} computing the idempotent function $f_{\text{blur}} : \Sigma_{\$}^* \rightarrow \Sigma_{\* , which satisfies $f_{\text{blur}}(\varepsilon) = \varepsilon$ and $f_{\text{blur}}(s_1 s_2 w) = s_1 s_1 f_{\text{blur}}(w)$ for any $s_1, s_2 \in \Sigma_{\$}$ and $w \in \Sigma_{\* .*

Proof. Let $T = (Q, \Sigma_{\$}, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a 1-tape Turing machine where

$$\begin{aligned}
Q &= \{q_{\text{ini}}, q_{\text{mov}}, q_{\text{read}}, q_{\text{back}}, q_{\text{pick}}, q_{\text{fin}}\} \cup \bigcup_{s \in \Sigma_{\$}} \{q_{\text{mem}(s)}, q_{\text{write}(s)}, q_{\text{keep}(s)}, q_{\text{check}(s)}\} \\
\Delta &= \{(q_{\text{ini}}, \sqcup \rightarrow \sqcup, q_{\text{mov}}), (q_{\text{mov}}, \rightarrow, q_{\text{read}}), (q_{\text{pick}}, \sqcup \rightarrow \sqcup, q_{\text{fin}}), (q_{\text{back}}, \leftarrow, q_{\text{pick}}), (q_{\text{read}}, \sqcup \rightarrow \sqcup, q_{\text{back}})\} \cup \\
&\quad \{(q_{\text{read}}, s \rightarrow s, q_{\text{mem}(s)}) \mid s \in \Sigma_{\$}\} \cup \{(q_{\text{mem}(s)}, \rightarrow, q_{\text{write}(s)}) \mid s \in \Sigma_{\$}\} \cup \\
&\quad \{(q_{\text{write}(s)}, s' \rightarrow s, q_{\text{mov}}) \mid s, s' \in \Sigma_{\$}\} \cup \{(q_{\text{pick}}, s \rightarrow s, q_{\text{keep}(s)}) \mid s \in \Sigma_{\$}\} \cup \\
&\quad \{(q_{\text{keep}(s)}, \leftarrow, q_{\text{check}(s)}) \mid s \in \Sigma_{\$}\} \cup \{(q_{\text{check}(s)}, s \rightarrow s, q_{\text{back}}) \mid s \in \Sigma_{\$}\}.
\end{aligned}$$

79:16 Idempotent Turing Machines

Then T is found to be idempotent by the set $Q' \subset Q$ of rear states and the rear state map ψ defined as

$$Q' = \{q_{\text{fin}}, q_{\text{pick}}, q_{\text{back}}\} \cup \bigcup_{s \in \Sigma_{\S}} \{q_{\text{check}(s)}, q_{\text{keep}(s)}\}$$

$$\psi = \{q_{\text{fin}} \mapsto q_{\text{ini}}, q_{\text{pick}} \mapsto q_{\text{mov}}, q_{\text{back}} \mapsto q_{\text{read}}\} \cup \bigcup_{s \in \Sigma_{\S}} \{q_{\text{check}(s)} \mapsto q_{\text{mem}(s)}, q_{\text{keep}(s)} \mapsto q_{\text{write}(s)}\}$$

since the four conditions obviously hold. We can check that T computes the function f_{blur} in the statement as follows. Firstly the relation

$$(q_{\text{read}}, \langle \varepsilon, s, ws'w' \rangle) \vdash^* (q_{\text{read}}, \langle \overleftarrow{f_{\text{blur}}(sw)}, s', w' \rangle)$$

holds for any $s, s' \in \Sigma_{\S}$ and $w, w' \in \Sigma_{\S}^*$ where the length of w is odd and \overleftarrow{x} denotes the reversed string of x , which can be proved by induction on the length of w . Moreover the relation

$$(q_{\text{back}}, \langle \overleftarrow{sw}, s', w' \rangle) \vdash^* (q_{\text{back}}, \langle \varepsilon, s, ws'w' \rangle)$$

holds for any $s, s' \in \Sigma_{\S}$ and $w, w' \in \Sigma_{\S}^*$ where the length of w and w' are odd and the strings sw and $s'w'$ are in the domain of f_{blur} , i.e., every even-numbered symbol in them is the same as the previous symbol. This can also be proved by induction on the length of w . Then, we have the relation

$$(q_{\text{ini}}, \langle \varepsilon, \sqcup, w \rangle) \vdash^* (q_{\text{fin}}, \langle \varepsilon, \sqcup, f_{\text{blur}}(w) \rangle)$$

whenever the length of w is even, which indicates that T computes f_{blur} . ◀

► **Theorem 23** (Robustness of idempotent Turing machines). *Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape idempotent Turing machine. Then there exists a 1-tape idempotent Turing machine T' such that $\llbracket T \rrbracket(w_1, \dots, w_k) = (v_1, \dots, v_k)$ if and only if $\llbracket T' \rrbracket(\text{enc}(w_1, \dots, w_k)) = \text{enc}(v_1, \dots, v_k)$.*

Proof. Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape idempotent Turing machine. Consider a function $f : \Sigma_{\S}^* \rightarrow \Sigma_{\S}^*$ satisfying

$$f(\text{enc}(w_1, \dots, w_k)) = \text{enc}(\text{enc}(\llbracket T \rrbracket(w_1, \dots, w_k)), \text{enc}(w_1, \dots, w_k))$$

for any $w_1, \dots, w_n \in \Sigma^*$ only if $\llbracket T \rrbracket(w_1, \dots, w_k)$ is defined. Since the function f is injective and computable, we can construct a 1-tape Turing machine T_f such that $\llbracket T_f \rrbracket = f$ by Theorem 15.

Let us define a 1-tape Turing machine $T' = T_f^{-1} \circ T_{\text{blur}} \circ T_f$ where the idempotent Turing machine T_{blur} is given by Lemma 22. Note that

$$f_{\text{blur}}(\text{enc}(x, y)) = \text{enc}(x, x) \tag{1}$$

holds for any $x, y \in \Sigma_{\S}^*$. Since T' is idempotent because of Lemma 19, it suffices to show T' simulates T under encoding with enc . This can be shown by

$$\begin{aligned} \llbracket T' \rrbracket(\text{enc}(w_1, \dots, w_k)) &= \{ \text{by the definition of } T' \text{ and Proposition 10} \} \\ &\llbracket T_f^{-1} \rrbracket(\llbracket T_{\text{blur}} \rrbracket(\llbracket T_f \rrbracket(\text{enc}(w_1, \dots, w_k)))) \end{aligned}$$

$$\begin{aligned}
&= \{ \text{by the semantics of } T_f \text{ and Proposition 14} \} \\
&\quad \llbracket T_f \rrbracket^{-1}(\llbracket T_{blur} \rrbracket(\text{enc}(\text{enc}(\llbracket T \rrbracket(w_1, \dots, w_k))), \text{enc}(w_1, \dots, w_k))) \\
&= \{ \text{by the semantics of } T_{blur} \text{ with Equation (1)} \} \\
&\quad \llbracket T_f \rrbracket^{-1}(\text{enc}(\text{enc}(\llbracket T \rrbracket(w_1, \dots, w_k))), \text{enc}(\llbracket T \rrbracket(w_1, \dots, w_k))) \\
&= \{ \text{by the idempotence of } T \text{ and Theorem 17} \} \\
&\quad \llbracket T_f \rrbracket^{-1}(\text{enc}(\text{enc}(\llbracket T \rrbracket(\llbracket T \rrbracket(w_1, \dots, w_k))), \text{enc}(\llbracket T \rrbracket(w_1, \dots, w_k)))) \\
&= \{ \text{by the injectivity of } f \text{ and the semantics of } f^{-1} \} \\
&\quad \text{enc}(\llbracket T \rrbracket(w_1, \dots, w_k)).
\end{aligned}$$

◀

► **Proposition 26** (Bennett's trick [3]). *Let T be a k -tape Turing machine. There exists a $(2k + 1)$ -tape reversible Turing machine $\text{Ben}_k(T)$ such that $\llbracket \text{Ben}_k(T) \rrbracket(w_1, \dots, w_k, \underbrace{\varepsilon, \dots, \varepsilon}_{k+1}) =$*

$(w_1, \dots, w_k, \llbracket T \rrbracket(w_1, \dots, w_k), \varepsilon)$ for any input w_1, \dots, w_k of T .

Proof. Let $T = (Q, \Sigma, q_{\text{ini}}, q_{\text{fin}}, \Delta)$ be a k -tape Turing machine. Firstly, by Landauer embedding [10], we can construct a $(k + 1)$ -tape Turing machine T_L such that

$$\llbracket T_L \rrbracket(w_1, \dots, w_k, \varepsilon) = (\llbracket T \rrbracket(w_1, \dots, w_k), \text{trace}(T, w_1, \dots, w_k))$$

where the function *trace* encodes the history of applied transition rules on the run into Σ^* . Since the history tells the previous configuration for each step, the Turing machine T_L is backward deterministic, that is, reversible. We extend T_L with k extra tapes in between working tapes and the history tape where the extra tapes are never touched during computation. Let T'_L be the $(2k + 1)$ -reversible Turing machine obtained by the extension. Then we have

$$\llbracket T'_L \rrbracket(w_1, \dots, w_k, v_1, \dots, v_k, \varepsilon) = (\llbracket T \rrbracket(w_1, \dots, w_k), v_1, \dots, v_k, \text{trace}(T, w_1, \dots, w_k))$$

for $w_1, \dots, w_k, v_1, \dots, v_k \in \Sigma^*$. In addition, we similarly extend the $2k$ -tape Turing machine $T_{\text{dup}(k)}$ with one extra tape to obtain $(2k + 1)$ -tape Turing machine $T'_{\text{dup}(k)}$ such that

$$\llbracket T'_{\text{dup}(k)} \rrbracket(w_1, \dots, w_k, \underbrace{\varepsilon, \dots, \varepsilon}_k, v) = (w_1, \dots, w_k, w_1, \dots, w_k, v)$$

holds for $w_1, \dots, w_k, v \in \Sigma^*$.

Let us define the $(2k + 1)$ -tape Turing machine $\text{Ben}_k(T) = T'^{-1}_L \circ T'_{\text{dup}(k)} \circ T'_L$ which is reversible because so are all constituents. Because of the domain of $T_{\text{dup}(k)}$ and the semantics of T'_L , the input of the Turing machine $\text{Ben}_k(T)$ is restricted to $(2k + 1)$ -tuples of strings whose $(k + 1)$ -th through $2k + 1$ -th strings are the empty string. Therefore the equation in the statement can be checked by

$$\begin{aligned}
&\llbracket \text{Ben}_k(T) \rrbracket(w_1, \dots, w_k, \underbrace{\varepsilon, \dots, \varepsilon}_k, \varepsilon) \\
&= \{ \text{by the definition of } \text{Ben}_k(T) \text{ and Proposition 10} \} \\
&\quad \llbracket T'^{-1}_L \rrbracket(\llbracket T'_{\text{dup}(k)} \rrbracket(\llbracket T'_L \rrbracket(w_1, \dots, w_k, \underbrace{\varepsilon, \dots, \varepsilon}_k, \varepsilon))) \\
&= \{ \text{by the semantics of } T'_L \text{ and Proposition 14} \} \\
&\quad \llbracket T'_L \rrbracket^{-1}(\llbracket T'_{\text{dup}(k)} \rrbracket(\llbracket T \rrbracket(w_1, \dots, w_k), \underbrace{\varepsilon, \dots, \varepsilon}_k, \text{trace}(T, w_1, \dots, w_k))))
\end{aligned}$$

79:18 Idempotent Turing Machines


$$\begin{aligned}
&= \{ \text{by the semantics of } T'_{dup(k)} \} \\
&\quad \llbracket T'_L \rrbracket^{-1}(\llbracket T \rrbracket(w_1, \dots, w_k), \llbracket T \rrbracket(w_1, \dots, w_k), \text{trace}(T, w_1, \dots, w_k)) \\
&= \{ \text{by the inverse of the semantics of } T'_L \} \\
&\quad (w_1, \dots, w_k, \llbracket T \rrbracket(w_1, \dots, w_k), \varepsilon). \quad \blacktriangleleft
\end{aligned}$$

► **Theorem 27** (IdTM-universal Turing machine constructed with Bennett's trick). *Let U be a (2-tape) classically universal Turing machine, i.e., $\llbracket U \rrbracket(\ulcorner T \urcorner, w) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w))$ for any Turing machine T and its input w . Then a 5-tape idempotent Turing machine $U' = \text{Ben}_2(U)^{-1} \circ T'_{copy(2)} \circ \text{Ben}_2(U)$ is IdTM-universal where $T'_{copy(2)}$ is obtained by adding one extra tape as the fifth tape to $T_{copy(2)}$, i.e., $\llbracket T'_{copy(2)} \rrbracket(w_1, w_2, w_3, w_4, w_5) = (w_3, w_4, w_3, w_4, w_5)$.*

Proof. Let U be a 2-tape classically universal Turing machine such that $\llbracket U \rrbracket(\ulcorner T \urcorner, w) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w))$ for any Turing machine T and its input w . Note that the Turing machine $U' = \text{Ben}_2(U)^{-1} \circ T'_{copy(2)} \circ \text{Ben}_2(U)$ is idempotent due to Lemma 19. We shall show the 5-tape idempotent Turing machine U' is IdTM-universal, that is, $\llbracket U' \rrbracket(\ulcorner T \urcorner, w, \varepsilon, \varepsilon, \varepsilon) = (\ulcorner T \urcorner, \llbracket T \rrbracket(w), \varepsilon, \varepsilon, \varepsilon)$ holds for any Turing machine T and its input w . Because of the domain of $\text{Ben}_2(U)$, the input of the Turing machine U' is restricted to 5-tuples whose third through fifth are the empty string. Therefore, the statement can be verified by

$$\begin{aligned}
\llbracket U' \rrbracket(\ulcorner T \urcorner, w, \varepsilon, \varepsilon, \varepsilon) &= \{ \text{by the definition of } U' \text{ and Proposition 10} \} \\
&\quad \llbracket \text{Ben}_2(U)^{-1} \rrbracket(\llbracket T'_{copy(2)} \rrbracket(\llbracket \text{Ben}_2(U) \rrbracket(\ulcorner T \urcorner, w, \varepsilon, \varepsilon, \varepsilon))) \\
&= \{ \text{by the semantics of } \text{Ben}_2(U) \text{ and } U \} \\
&\quad \llbracket \text{Ben}_2(U)^{-1} \rrbracket(\llbracket T'_{copy(2)} \rrbracket(\ulcorner T \urcorner, w, \ulcorner T \urcorner, \llbracket T \rrbracket(w), \varepsilon)) \\
&= \{ \text{by the semantics of } T'_{copy(2)} \} \\
&\quad \llbracket \text{Ben}_2(U)^{-1} \rrbracket(\ulcorner T \urcorner, \llbracket T \rrbracket(w), \ulcorner T \urcorner, \llbracket T \rrbracket(w), \varepsilon) \\
&= \{ \text{by the idempotence of } T \text{ and Theorem 17} \} \\
&\quad \llbracket \text{Ben}_2(U)^{-1} \rrbracket(\ulcorner T \urcorner, \llbracket T \rrbracket(w), \ulcorner T \urcorner, \llbracket T \rrbracket(\llbracket T \rrbracket(w)), \varepsilon) \\
&= \{ \text{by the inverse of the semantics of } \text{Ben}_2(U) \} \\
&\quad (\ulcorner T \urcorner, \llbracket T \rrbracket(w), \varepsilon, \varepsilon, \varepsilon). \quad \blacktriangleleft
\end{aligned}$$

Ergodic Theorems and Converses for PSPACE Functions

Satyadev Nandakumar ✉ 

Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, India

Subin Pulari ✉ 

Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, India

Abstract

We initiate the study of effective pointwise ergodic theorems in resource-bounded settings. Classically, the convergence of the ergodic averages for integrable functions can be arbitrarily slow [14]. In contrast, we show that for a class of PSPACE L^1 functions, and a class of PSPACE computable measure-preserving ergodic transformations, the ergodic average exists and is equal to the space average on every EXP random. We establish a partial converse that PSPACE non-randomness can be characterized as non-convergence of ergodic averages. Further, we prove that there is a class of resource-bounded randoms, *viz.* SUBEXP-space randoms, on which the corresponding ergodic theorem has an exact converse - a point x is SUBEXP-space random if and only if the corresponding effective ergodic theorem holds for x .

2012 ACM Subject Classification Theory of computation → Constructive mathematics; Theory of computation → Complexity theory and logic; Mathematics of computing → Probability and statistics

Keywords and phrases Ergodic Theorem, Resource-bounded randomness, Computable analysis, Complexity theory

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.80

Related Version *Full Version:* <https://arxiv.org/abs/2012.11266>

Acknowledgements The authors wish to thank anonymous reviewers for helpful suggestions.

1 Introduction

In Kolmogorov’s program to found information theory on the theory of algorithms, we investigate whether individual “random” objects obey probabilistic laws, *i.e.*, properties which hold in sample spaces with probability 1. Indeed, a vast and growing literature establishes that *every* Martin-Löf random sequence (see for example, [4] or [19]) obeys the Strong Law of Large Numbers [24], the Law of Iterated Logarithm [25], and surprisingly, the Birkhoff Ergodic Theorem [26, 17, 10, 1] and the Shannon-McMillan-Breiman theorem [8, 9, 21]. In effective settings, the theorem for Martin-Löf random points implies the classical theorem since the set of Martin-Löf randoms has Lebesgue measure 1, and hence is stronger.

In this work, we initiate the study of ergodic theorems in resource-bounded settings. This is a difficult problem, since classically, the convergence speed in ergodic theorems is known to be arbitrarily slow (e.g. see Bishop [3], Krengel [14], and V’yugin [26]). However, we establish ergodic theorems in resource-bounded settings which hold on every resource-bounded random object of a particular class. The main technical hurdle we overcome is the lack of sharp tail bounds. The only general tail bound in ergodic settings is the maximal ergodic inequality. This yields only an inverse linear bound in the error bound, in contrast to the inverse exponential bounds in the Chernoff and the Azuma-Hoeffding inequalities.



© Satyadev Nandakumar and Subin Pulari;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 80; pp. 80:1–80:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We first establish an unconditional result. – For the entire class of PSPACE L^1 functions on Bernoulli systems, the ergodic average exists and is equal to the space average on all EXP randoms. We utilize a non-trivial connection with the theory of uniform distribution of sequences modulo 1 [15, 16, 20, 18] to prove this result.

In the general case, rapid L^1 convergence of subsequences of ergodic averages suffices to establish the same consequence that the ergodic average exists and is equal to the space average on all EXP randoms. In general, such assumptions are unavoidable since an adaptation of V’yugin’s counterexample [26] shows that there are PSPACE computable ergodic Markov systems where the convergence rate to the ergodic average is not even computable.

Conversely, we ask whether we can characterize non-randomness using the failure of the PSPACE ergodic theorem. Franklin and Towsner [5] show that for every non-Martin-Löf random x , there is an effective ergodic system where the ergodic average at x does not converge to the space average. We first show that our PSPACE effective ergodic theorem admits a partial converse of this form. PSPACE non-randoms can be characterized as points where the PSPACE ergodic theorem fails.

We know that the set of EXP randoms is a subset of the set of PSPACE randoms. Since the forward direction holds on the smaller set of randoms, it is important to know whether there is a class of resource-bounded randoms on which an effective ergodic theorem holds with an exact converse. We show that the class of SUBEXP-space randoms is one such. We summarize our results in Table 1.

The proofs of these results are adapted from the techniques of Rute [21], Ko [13], Galatolo, Hoyrup & Rojas [7, 11], and Huang & Stull [12].¹ Our proofs involve several new quantitative estimates, which may of general interest.

■ **Table 1** Summary of the results involving PSPACE/SUBEXP-space systems.

Class of functions	Convergence of ergodic averages (Theorems)	
	$\forall f(A_n^f \rightarrow \int f d\mu)$	$\exists f(A_n^f \not\rightarrow \int f d\mu)$
PSPACE L^1	EXP randoms (6.2)	PSPACE nonrandoms (7.1)
SUBEXP-space L^1	SUBEXP-space randoms (8.11)	SUBEXP-space nonrandoms (8.12)

2 Preliminaries

Let $\Sigma = \{0, 1\}$ be the binary alphabet. Denote the set of all finite binary strings by Σ^* and the set of infinite binary strings by Σ^∞ . For $\sigma \in \Sigma^*$ and $y \in \Sigma^* \cup \Sigma^\infty$, we write $\sigma \sqsubseteq y$ if σ is a prefix of y . For any infinite string y and any finite string σ , $\sigma[n]$ and $y[n]$ denotes the character at the n^{th} position in y and σ respectively. For any infinite string y and any finite string σ , $\sigma[n, m]$ and $y[n, m]$ represents the strings $\sigma[n]\sigma[n + 1] \dots \sigma[m]$ and $y[n]y[n + 1] \dots y[m]$ respectively. We denote finite strings using small Greek letters like σ , α etc. The length of a finite binary string σ is denoted by $|\sigma|$.

¹ There are alternative approaches to the proof in Martin-Löf settings, like that of V’yugin [26]. However, the tool he uses for establishing the result is a lower semicomputable test defined on infinite sequences - this is difficult to adapt to resource-bounded settings requiring the output value within bounded time or space. Moreover, the functions in V’yugin’s approach are continuous. We consider the larger class of L^1 functions, which can be discontinuous in general.

For $\sigma \in \Sigma^*$, the *cylinder* $[\sigma]$ is the set of all infinite sequences with σ as a prefix. χ_σ denotes the characteristic function of $[\sigma]$. For any set of strings $S \subseteq \Sigma^*$, $[S]$ is the union of $[\sigma]$ over all $\sigma \in S$. Extending the notation, χ_S denotes the characteristic function of $[S]$. The Borel σ -algebra generated by the set of all cylinders is denoted by $\mathcal{B}(\Sigma^\infty)$.

Unless specified otherwise, any $n \in \mathbb{N}$ is represented in the binary alphabet. As is typical in resource-bounded settings, some integer parameters are represented in unary. The set of unary strings is represented as 1^* , and the representation of $n \in \mathbb{N}$ in unary is 1^n , a string consisting of n ones. For any $n_1, n_2 \in \mathbb{N}$, $[n_1, n_2]$ represents the set $\{n \in \mathbb{N} : n_1 \leq n \leq n_2\}$.

Throughout the paper we take into account the number of cells used in the output tape and the working tape when calculating the space complexity of functions. We assume a finite representation for the set of rational numbers \mathbb{Q} satisfying the following: there exists a $c \in \mathbb{N}$ such that if $r \in \mathbb{Q}$ has a representation of length l then $r \leq 2^{lc}$. Following the works of Hoyrup, and Rojas [11], we introduce the notion of a PSPACE-probability Cantor space by endowing the Cantor space with a PSPACE-computable probability measure.

► **Definition 2.1.** *Consider the Cantor space $(\Sigma^\infty, \mathcal{B}(\Sigma^\infty))$. A Borel probability measure $\mu : \mathcal{B}(\Sigma^\infty) \rightarrow [0, 1]$, is a PSPACE-probability measure if there is a PSPACE machine $M : \Sigma^* \times 1^* \rightarrow \mathbb{Q}$ such that for every $\sigma \in \Sigma^*$, and $n \in \mathbb{N}$, we have that $|M(\sigma, 1^n) - \mu([\sigma])| \leq 2^{-n}$.*

In order to define PSPACE (EXP) randomness using PSPACE (EXP) tests we require the following method for approximating sequences of open sets in Σ^∞ in polynomial space (exponential time).

► **Definition 2.2** (PSPACE/EXP sequence of open sets [12]). *A sequence of open sets $\langle U_n \rangle_{n=1}^\infty$ is a PSPACE sequence of open sets if there exists a sequence of sets $\langle S_n^k \rangle_{k,n \in \mathbb{N}}$ where $S_n^k \subseteq \Sigma^*$ such that*

1. $U_n = \bigcup_{k=1}^\infty [S_n^k]$, where for any $m > 0$, $\mu(U_n - \bigcup_{k=1}^m [S_n^k]) \leq 2^{-m}$.
2. There exists a controlling polynomial p such that $\max\{|\sigma| : \sigma \in \bigcup_{k=1}^m S_n^k\} \leq p(n+m)$.
3. The function $g : \Sigma^* \times 1^* \times 1^* \rightarrow \{0, 1\}$ such that $g(\sigma, 1^n, 1^m) = 1$ if $\sigma \in S_n^m$, and 0 otherwise, is decidable by a PSPACE machine.

The definition of EXP sequence of open sets is similar but the bound in condition 2 is replaced with $2^{p(n+m)}$ and the machine in condition 3 is an EXP-time machine.

Henceforth, we study the notion of resource bounded randomness on (Σ^∞, μ) .

► **Definition 2.3** (PSPACE/EXP randomness [23]). *A sequence of open sets $\langle U_n \rangle_{n=1}^\infty$ is a PSPACE test if it is a PSPACE sequence of open sets and for all $n \in \mathbb{N}$, $\mu(U_n) \leq 2^{-n}$.*

A set $A \subseteq \Sigma^\infty$ is PSPACE null or PSPACE non-random if there is a PSPACE test $\langle U_n \rangle_{n=1}^\infty$ such that $A \subseteq \bigcap_{n=1}^\infty U_n$, and is PSPACE random otherwise. The EXP analogues of the above concepts are defined similarly except that $\langle U_n \rangle_{n=1}^\infty$ is an EXP sequence of open sets.

By considering the sequence $\langle \bigcup_{i=1}^k S_n^i \rangle_{k,n \in \mathbb{N}}$ instead of $\langle S_n^k \rangle_{k,n \in \mathbb{N}}$, without loss of generality, we can assume that for each n , $\langle S_n^k \rangle_{k=1}^\infty$ is an increasing sequence of sets. Since every PSPACE test is an EXP test, every EXP random is PSPACE random.

In order to establish our ergodic theorem, it is convenient to define a PSPACE version of Solovay tests, where the relaxation is that the measures of the sets U_n can be any sufficiently fast convergent sequence. We later show that this captures the same set of randoms as PSPACE tests.

► **Definition 2.4** (PSPACE Solovay test). *A sequence of open sets $\langle U_n \rangle_{n=1}^\infty$ is a PSPACE Solovay test if it is a PSPACE sequence of open sets and there is a polynomial p such that $\sum_{n=p(m)+1}^\infty \mu(U_n) \leq 2^{-m}$ for all $m \in \mathbb{N} \setminus \{0\}$.² A set $A \subseteq \Sigma^\infty$ is PSPACE Solovay null or PSPACE Solovay non-random if there exists a PSPACE Solovay test $\langle U_n \rangle_{n=1}^\infty$ such that $A \subseteq \bigcap_{i=1}^\infty \bigcup_{n=i}^\infty U_n$, and is PSPACE Solovay random otherwise.*

► **Theorem 2.5.** *A set $A \subseteq \Sigma^\infty$ is PSPACE null if and only if A is PSPACE Solovay null.*

The set of PSPACE Solovay randoms and PSPACE randoms are equal, hence to prove PSPACE randomness results, it suffices to form Solovay tests.

3 PSPACE L^1 computability

The resource-bounded ergodic theorems in our work hold for PSPACE- L^1 functions, the PSPACE analogue of integrable functions. In this section, we briefly recall standard definitions for PSPACE computable L^1 functions and measure-preserving transformations. The justifications and proofs of equivalences of various notions are present in Stull's thesis [22] and [23]. We initially define PSPACE sequence of simple functions, and define PSPACE integrable functions based on approximations using these functions.

► **Definition 3.1** (PSPACE sequence of simple functions [23]). *A sequence of simple functions $\langle f_n \rangle_{n=1}^\infty$, where each $f_n : \Sigma^\infty \rightarrow \mathbb{Q}$, is a PSPACE sequence of simple functions if*

1. *There is a controlling polynomial p such that for each n , there exists $k(n) \in \mathbb{N}$, $\{d_1, d_2, \dots, d_{k(n)}\} \subseteq \mathbb{Q}$ and $\{\sigma_1, \sigma_2, \dots, \sigma_{k(n)}\} \subseteq \Sigma^{p(n)}$ satisfying $f_n = \sum_{i=1}^{k(n)} d_i \chi_{\sigma_i}$.*
2. *There is a PSPACE machine M such that for each $n \in \mathbb{N}$, and $\sigma \in \Sigma^*$, $M(1^n, \sigma)$ outputs $f_n(\sigma 0^\infty)$ if $|\sigma| \geq p(n)$ and ? otherwise.*

Note that since M is a PSPACE machine, $\{d_1, d_2 \dots d_{k(n)}\}$ is a set of PSPACE representable numbers. Now, we define PSPACE L^1 -computable functions in terms of limits of convergent PSPACE sequence of simple functions.

► **Definition 3.2** (PSPACE L^1 -computable functions [23]). *A function $f \in L^1(\Sigma^\infty, \mu)$ is PSPACE L^1 -computable if there exists a PSPACE sequence of simple functions $\langle f_n \rangle_{n=1}^\infty$ such that for every $n \in \mathbb{N}$, $\|f - f_n\| \leq 2^{-n}$. The sequence $\langle f_n \rangle_{n=1}^\infty$ is called a PSPACE L^1 -approximation of f .*

A sequence of L^1 functions $\langle f_n \rangle_{n=1}^\infty$ converging to f in the L^1 -norm need not have pointwise limits. Hence the following concept ([21]) is important in studying the pointwise ergodic theorem in the setting of L^1 -computability

► **Definition 3.3** (\tilde{f} for PSPACE L^1 -computable f). *Let $f \in L^1(\Sigma^\infty, \mu)$ be PSPACE L^1 -computable and with a PSPACE L^1 approximation $\langle f_n \rangle_{n=1}^\infty$. Define $\tilde{f} : \Sigma^\infty \rightarrow \mathbb{R} \cup \{\text{undefined}\}$ by $\tilde{f}(x) = \lim_{n \rightarrow \infty} f_n(x)$ if this limit exists, and is undefined otherwise.³*

To define ergodic averages, we restrict ourselves to the following class of transformations.

² This implies that $\sum_{n=1}^\infty \mu(U_n) < \infty$.

³ The definition of \tilde{f} is dependent on the choice of the approximating sequence $\langle f_n \rangle_{n=1}^\infty$. However, due to Lemma 4.3, we use \tilde{f} in a sequence independent manner.

► **Definition 3.4** (PSPACE simple transformation). *A measurable function $T : (\Sigma^\infty, \mu) \rightarrow (\Sigma^\infty, \mu)$ is a PSPACE simple transformation if there is a controlling constant c and a PSPACE machine M such that for any $\sigma \in \Sigma^*$, $T^{-1}([\sigma]) = \cup_{i=1}^{k(\sigma)} [\sigma_i]$ where the following properties hold.*

1. $\{\sigma_i\}_{i=1}^{k(\sigma)}$ is a prefix free set and for all $1 \leq i \leq k(\sigma)$, $|\sigma_i| \leq |\sigma| + c$
2. For each $\sigma, \alpha \in \Sigma^*$,

$$M(\sigma, \alpha) = \begin{cases} 1 & \text{if } |\alpha| \geq |\sigma| + c \text{ and } \alpha 0^\infty \in T^{-1}([\sigma]) \\ 0 & \text{if } |\alpha| \geq |\sigma| + c \text{ and } \alpha 0^\infty \notin T^{-1}([\sigma]) \\ ? & \text{otherwise} \end{cases}$$

PSPACE computability as defined above, relates naturally to convergence of L^1 norms. But the pointwise ergodic theorem deals with almost everywhere convergence, and its resource-bounded versions deal with convergence on every random point. We introduce the modes of convergence we deal with in the present work.

► **Definition 3.5** (PSPACE-rapid limit point). *A real number a is a PSPACE-rapid limit point of the real number sequence $\langle a_n \rangle_{n=1}^\infty$ if there exists a polynomial p such that for all $m \in \mathbb{N}$, $\exists k \leq 2^{p(m)}$ such that $|a_k - a| \leq 2^{-m}$.*

Note that this requires rapid convergence only on a subsequence, which may not be a computable subsequence of the full sequence. The following definition is the L^1 version of the above.

► **Definition 3.6** (PSPACE-rapid L^1 -limit point). *A function $f \in L^1(\Sigma^\infty, \mu)$ is a PSPACE-rapid L^1 -limit point of a sequence $\langle f_n \rangle_{n=1}^\infty$ of functions in $L^1(\Sigma^\infty, \mu)$ if 0 is a PSPACE-rapid limit point of $\|f_n - f\|_1$.*

Now we define PSPACE analogue of almost everywhere convergence ([21]).

► **Definition 3.7** (PSPACE-rapid almost everywhere convergence). *A sequence of measurable functions $\langle f_n \rangle_{n=1}^\infty$ is PSPACE-rapid almost everywhere convergent to a measurable function f if there exists a polynomial p such that for all m_1 and m_2 ,*

$$\mu \left(\left\{ x : \sup_{n \geq 2^{p(m_1+m_2)}} |f_n(x) - f(x)| \geq 2^{-m_1} \right\} \right) \leq 2^{-m_2}.$$

Notation. Let $A_n^{f,T} = \frac{f + f \circ T + f \circ T^2 + \dots + f \circ T^{n-1}}{n}$ denote the n^{th} Birkhoff average for any function f and transformation T . We prove the ergodic theorem in measure preserving systems where $\int f d\mu$ is a PSPACE-rapid L^1 -limit point of $A_n^{f,T}$. In the rest of the paper we denote $A_n^{f,T}$ simply by A_n^f . The transformation T involved in the Birkhoff sum is implicit.

PSPACE rapidity of A_n^f is a stronger version of \ln^2 -ergodicity introduced in [6].

► **Lemma 3.8.** *Let $T : \Sigma^\infty \rightarrow \Sigma^\infty$ be any measurable transformation and $f \in L^\infty(\Sigma^\infty, \mu)$. $\int f d\mu$ is a PSPACE-rapid L^1 -limit point of A_n^f if and only if there exists $c > 0$ and $k \in \mathbb{N}$ such that for all $n > 0$,*

$$\left| \frac{1}{n} \sum_{i=0}^{n-1} \int f \circ T^i \cdot f - \left(\int f \right)^2 d\mu \right| \leq \frac{c}{2^{(\ln n)^{\frac{1}{k}}}}.$$

4 PSPACE-rapid almost everywhere convergence of ergodic averages

We present PSPACE versions of Theorem 2 and Proposition 5 from [7], relating the L^1 convergence of A_n^f to $\int f$ to its almost everywhere convergence. The main estimate which we require in this section is the maximal ergodic inequality, which we now recall.

► **Lemma 4.1** (Maximal ergodic inequality [2]). *If $f \in L^1(\Sigma^\infty, \mu)$ and $\delta > 0$ then $\mu(\{x : \sup_{n \geq 1} |A_n^f(x)| > \delta\}) \leq (\|f\|_1)\delta^{-1}$.*

Using this lemma, we now prove the almost everywhere convergence of ergodic averages.

► **Theorem 4.2.** *Let f be any function in $L^1(\Sigma^\infty, \mu)$ and let T be a measure preserving transformation. If $\int f d\mu$ is a PSPACE-rapid L^1 -limit point of A_n^f then A_n^f is PSPACE-rapid almost everywhere convergent to $\int f d\mu$.*

If $f \in L^\infty$, the converse of Theorem 4.2 can be easily obtained by expanding $\|A_n^f - \int f d\mu\|_1$.

Now, we prove some auxiliary results that are useful in the proof of the PSPACE ergodic theorem. The following fact was shown in [12]. However, for our ergodic theorem we require an alternate proof of this fact using techniques from [21].

► **Lemma 4.3.** *Let $\langle f_n \rangle_{n=1}^\infty, \langle g_n \rangle_{n=1}^\infty$ be PSPACE sequence of simple functions which converges PSPACE-rapid almost everywhere to $f \in L^1(\Sigma^\infty, \mu)$. Then, for all EXP random x , $\lim_{n \rightarrow \infty} f_n(x)$ and $\lim_{n \rightarrow \infty} g_n(x)$ exist, and are equal.*

The following immediately follows from the above lemma.

► **Corollary 4.4.** *Let $f \in L^1(\Sigma^\infty, \mu)$ be a PSPACE L^1 -computable function with L^1 approximating PSPACE sequences of simple functions $\langle f_n \rangle_{n=1}^\infty$ and $\langle g_n \rangle_{n=1}^\infty$. Then, for all EXP random x $\lim_{n \rightarrow \infty} f_n(x)$ and $\lim_{n \rightarrow \infty} g_n(x)$ exist, and are equal.*

The following properties satisfied by PSPACE simple transformations and PSPACE L^1 -computable functions are useful in our proof of the PSPACE ergodic theorem.

► **Lemma 4.5.** *Let f be a PSPACE L^1 -computable function over the Bernoulli space. Let $I_f : \Sigma^\infty \rightarrow \Sigma^\infty$ be the constant function taking the value $\int f d\mu$ over all $x \in \Sigma^\infty$. Then, I_f is PSPACE L^1 -computable and $I_f(x) = \int f d\mu$ for all EXP random x .*

► **Lemma 4.6.** *Let f be a PSPACE L^1 -computable function with an L^1 approximating PSPACE sequence of simple functions $\langle f_n \rangle_{n=1}^\infty$. Let T be a PSPACE simple transformation and p be a polynomial. Then, $\langle A_n^{f_{p(n)}} \rangle_{n=1}^\infty$ is a PSPACE sequence of simple functions.*

5 Unconditional PSPACE ergodic theorem for the Bernoulli space

We now prove an unconditional version of our main result, namely, that for PSPACE L^1 computable functions, the ergodic average exists, and is equal to the space average, on every EXP random in the canonical setting of the Bernoulli space. We utilize the almost everywhere convergence results proved in the previous section, to prove the convergence on every PSPACE/EXP random. We first show that in the Bernoulli space, every PSPACE L^1 function exhibits PSPACE rapidity of A_n^f . The proof of this theorem is a non-trivial application of techniques from uniform distribution of sequences modulo 1 [15, 20, 16, 18].

► **Theorem 5.1.** *Let $f \in L^1(\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu)$ where μ is the Bernoulli measure $\mu(\sigma) = \frac{1}{2^{|\sigma|}}$ and let T be the left shift transformation. If f is PSPACE L^1 -computable, then there exists a polynomial q satisfying the following: given any $m \in \mathbb{N}$, for all $n \geq 2^{q(m)}$, $\|A_n^f - \int f d\mu\|_1 \leq 2^{-m}$.*

An equivalent statement is the following: The left-shift transformation on the Bernoulli probability measure is PSPACE ergodic⁴. Theorem 5.1 gives an explicit bound on the speed of convergence in the L^1 ergodic theorem for an interesting class of functions over the Bernoulli space. Such bounds do not exist in general for the L^1 ergodic theorem as demonstrated by Krengel in [14].

The above theorem can be obtained from the following assertion regarding PSPACE-rapid convergence of characteristic functions of long enough cylinders.

► **Lemma 5.2.** *Let T be the left shift transformation $T : (\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu) \rightarrow (\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu)$ where μ is the Bernoulli measure $\mu(\sigma) = 2^{-|\sigma|}$. There exist polynomials q_1, q_2 such that for any $m \in \mathbb{N}$ and $\sigma \in \Sigma^*$ with $|\sigma| \geq q_1(m)$ we get $\|A_n^{\chi_\sigma} - \mu(\sigma)\|_1 \leq 2^{-m}$ for all $n \geq |\sigma|^{3 \cdot 2^{q_2(m)}}$.*

Proof sketch. The major difficulty in directly approximating $\|A_n^{\chi_\sigma} - \mu(\sigma)\|_1$ is that for any $n, m \in \mathbb{N}$, $A_n^{\chi_\sigma}$ and $A_m^{\chi_\sigma}$ may not be *independent*. In order to overcome this, we use constructions similar to those used in proving Pillai's theorem (see [20], [16] for normal numbers, [18] for continued fractions) in order to approximate each $A_n^{\chi_\sigma}$ with sums of *disjoint* averages as follows.

$$A_n^{\chi_\sigma}(x) = \frac{\sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} X_i^{1,1}(x)}{n} + \sum_{p=2}^{\lfloor \log_2(\frac{n}{k}) \rfloor} \sum_{j=1}^{k-1} \frac{\sum_{i=1}^{\lfloor \frac{n}{2^{p-1}k} \rfloor} X_i^{p,j}}{n} + \frac{(k-1) \cdot O(\log n)}{n}, \quad \text{where}$$

$$X_i^{1,1}(x) = \begin{cases} 1 & \text{if } x[ik+1, (i+1)k] = \sigma \\ 0 & \text{otherwise,} \end{cases} \quad \text{and}$$

$$X_i^{p,j}(x) = \begin{cases} 1 & \text{if } x[2^{p-2}k-j+1, 2^{p-2}k-j+k] = \sigma \\ 0 & \text{otherwise} \end{cases}$$

The first two terms on the right of the equation turns out to be averages of independent Bernoulli random variables. Hence, elementary results from probability theory regarding independent Bernoulli random variables can be used to show that $A_n^{\chi_\sigma}$ converges to $\int f d\mu$ sufficiently fast. ◀

We remark that since Lemma 5.2 is true with the L^1 -norm replaced by the L^2 -norm, Theorem 5.1 is also true in the L^2 setting. i.e, if a function f is PSPACE L^2 -computable (replacing L^1 norms with L^2 norms in definition 3.2) then there exists a polynomial q satisfying the following: given any $m \in \mathbb{N}$, for all $n \geq 2^{q(m)}$, $\|A_n^f - \int f d\mu\|_2 \leq 2^{-m}$. Hence, for PSPACE L^2 -computable functions and the left shift transformation T , we get bounds on the convergence speed in the von-Neumann's ergodic theorem.

It is easy to verify that if T is a PSPACE simple transformation then for any $n \geq 2$, T^n is also a PSPACE simple transformation. We need the following stronger assertion in the proof of the ergodic theorem.

⁴ Equivalently, there exists a constant c such that for all $n > 0$, $\|A_n^f - \int f d\mu\|_1 \leq 2^{-\lfloor \log(n)^{\frac{1}{c}} \rfloor}$.

► **Lemma 5.3.** *Let $T : (\Sigma^\infty, \mu) \rightarrow (\Sigma^\infty, \mu)$ be a PSPACE simple transformation with controlling constant c . There exists a PSPACE machine N such that for each $n \in \mathbb{N}$ and $\sigma, \alpha \in \Sigma^*$,*

$$N(1^n, \sigma, \alpha) = \begin{cases} 1 & \text{if } |\alpha| \geq |\sigma| + cn \text{ and } \alpha 0^\infty \in T^{-n}([\sigma]) \\ 0 & \text{if } |\alpha| \geq |\sigma| + cn \text{ and } \alpha 0^\infty \notin T^{-n}([\sigma]) \\ ? & \text{otherwise} \end{cases}$$

Proof of Lemma 5.3. Let M be the machine witnessing the fact that T is a PSPACE simple transformation with the polynomial space complexity bound $p(n)$. Let the machine N do the following on input $(1^n, \sigma, \alpha)$:

1. If $\alpha < |\sigma| + cn$, then output ?.
2. If $n = 1$ then, run $M(\sigma, \alpha)$ and output the result of this simulation.
3. Else:
 - a. For all strings α' of length $|\sigma| + c(n-1)$ do the following:
 - i. If $N(1^{n-1}, \sigma, \alpha') = 1$ then, output 1 if $M(\alpha', \alpha) = 1$.
4. If no output is produced in the above steps, output 0.

When $n = 1$, N uses at most $p(|\sigma| + |\alpha| + cn) + O(1)$ space. Inductively, assume that for $n = k$, N uses at most $(2k-1)p(|\sigma| + |\alpha| + cn) + O(1)$ space. For $n = k+1$, the storage of α' and the two simulations inside step 3a can be done in $2p(|\sigma| + |\alpha| + cn) + (2k-1)p(|\sigma| + |\alpha| + cn) + O(1) = (2(k+1) - 1)p(|\sigma| + |\alpha| + cn) + O(1)$ space. Hence, N is a PSPACE machine. ◀

Now, we prove the unconditional ergodic theorem for PSPACE L^1 functions over the Bernoulli space. The proof involves adaptations of techniques from Rute [21], together with new quantitative bounds which yield the result within prescribed resource bounds.

► **Theorem 5.4.** *Let T be the left shift transformation $T : (\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu) \rightarrow (\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu)$ where μ is the Bernoulli measure $\mu(\sigma) = 2^{-|\sigma|}$. Then, for any PSPACE L^1 -computable f ,*

$$\lim_{n \rightarrow \infty} \widetilde{A}_n^f = \int f d\mu \text{ on EXP randoms.}$$

Proof of Theorem 5.4. Let $\langle f_m \rangle_{m=1}^\infty$ be any PSPACE sequence of simple functions L^1 approximating f . We initially approximate A_n^f with a PSPACE sequence of simple functions $\langle g_n \rangle_{n=1}^\infty$ which converges to $\int f d\mu$ on EXP randoms. Then we show that \widetilde{A}_n^f has the same limit as g_n on PSPACE randoms and hence on EXP randoms.

For each n , it is easy to verify that $\langle A_n^{f_m} \rangle_{m=1}^\infty$ is a PSPACE sequence of simple functions L^1 approximating A_n^f with the same rate of convergence. Using techniques similar to those in Lemma 4.3 and Corollary 4.4, we can obtain a polynomial p such that

$$\mu \left(\left\{ x : \sup_{m \geq p(n+i)} |A_n^{f_m}(x) - A_n^{f_{p(n+i)}}(x)| \geq \frac{1}{2^{n+i+1}} \right\} \right) \leq \frac{1}{2^{n+i+1}}.$$

For every $n > 0$, let $g_n = A_n^{f_{p(n)}}$. We initially show that $\langle g_n \rangle_{n=1}^\infty$ converges to $\int f d\mu$ on EXP randoms. Let $m_1, m_2 \geq 0$. From Theorem 4.2, A_n^f is PSPACE-rapid almost everywhere convergent to $\int f d\mu$. Hence there is a polynomial q such that

$$\mu \left(\left\{ x : \sup_{n \geq 2^{q(m_1+m_2)}} |A_n^f(x) - \int f d\mu| \geq \frac{1}{2^{m_1+1}} \right\} \right) \leq \frac{1}{2^{m_2+1}}.$$

Let $N(m_1, m_2) = \max\{2m_1, 2m_2, 2^{q(m_1+m_2)}\}$. Then,

$$\sum_{n \geq N(m_1, m_2)} \frac{1}{2^{k+1}} = \frac{1}{2^{N(m_1, m_2)}} \leq \min \left\{ \frac{1}{2^{m_1+1}}, \frac{1}{2^{m_2+1}} \right\}.$$

Let

$$G_n = \left\{ x : \sup_{n \geq N(m_1, m_2)} |g_n - \int f d\mu| > \frac{1}{2^{m_1}} \right\}.$$

Now, we have

$$\begin{aligned} \mu(G_n) &\leq \sum_{n \geq N(m_1, m_2)} \mu \left(\left\{ x : |g_n - A_n^f(x)| > \frac{1}{2^{m_1+1}} \right\} \right) \\ &\quad + \mu \left(\left\{ x : \sup_{n \geq 2^{q(m_1+m_2)}} |A_n^f(x) - \int f d\mu| \geq \frac{1}{2^{m_1+1}} \right\} \right) \\ &\leq \sum_{n \geq N(m_1, m_2)} \frac{1}{2^{n+1}} + \frac{1}{2^{m_2+1}} \\ &\leq \frac{1}{2^{m_2}}. \end{aligned}$$

Note that $N(m_1, m_2)$ is bounded by $2^{(m_1+m_2)^c}$ for some $c \in \mathbb{N}$. Hence, g_n is PSPACE-rapid almost everywhere convergent to $\int f d\mu$. From Lemma 4.6 it follows that $\langle g_n \rangle_{n=1}^\infty = \langle A_n^{f_{p(n)}} \rangle_{n=1}^\infty$ is a PSPACE sequence of simple functions (in parameter n). Let $I_f : \Sigma^\infty \rightarrow \Sigma^\infty$ be the constant function taking the value $\int f d\mu$ over all $x \in \Sigma^\infty$. From the above observations and Lemma 4.3 we get that $\lim_{n \rightarrow \infty} g_n(x) = I_f(x)$ for any x which is EXP random. From Lemma 4.5, we get that $\lim_{n \rightarrow \infty} g_n(x) = \int f d\mu$ for any x which is EXP random.

We now show that $\lim_{n \rightarrow \infty} \tilde{A}_n^f = \lim_{n \rightarrow \infty} g_n$ on PSPACE randoms. Define

$$U_{n,i} = \left\{ x : \max_{p(n+i) \leq m \leq p(n+i+1)} |A_n^f(x) - A_n^{f_{p(n+i)}}(x)| \geq \frac{1}{2^{n+i+1}} \right\}.$$

We already know $\mu(U_{n,i}) \leq \frac{1}{2^{n+i+1}}$. $U_{n,i}$ can be shown to be polynomial space approximable in parameters n and i in the following sense. There exists a sequence of sets of strings $\langle S_{n,i} \rangle_{i,n \in \mathbb{N}}$ and polynomial p satisfying the following conditions:

1. $U_{n,i} = [S_{n,i}]$.
2. There exists a *controlling polynomial* r such that $\max\{|\sigma| : \sigma \in S_{n,i}\} \leq r(n+i)$.
3. The function $g : \Sigma^* \times 1^* \times 1^* \rightarrow \{0, 1\}$ such that

$$g(\sigma, 1^n, 1^i) = \begin{cases} 1 & \text{if } \sigma \in S_{n,i} \\ 0 & \text{otherwise,} \end{cases}$$

is decidable by a PSPACE machine.

The above claims can be established by using techniques similar to those in Lemma 4.6 and Lemma 4.3. We show the construction of a machine N computing the function g above. Let M_f be a computing machine and let q be a controlling polynomial for $\langle f_n \rangle_{n=1}^\infty$. Let c be a controlling constant for T . Let M' be the machine from Lemma 5.3. Machine N on input $(\sigma, 1^n, 1^i)$ does the following:

80:10 Ergodic Theorems and Converses for PSPACE Functions

1. If $|\sigma| > q(p(n+i+1)) + cn$, then output 0.
2. Compute $A_n^{f_{p(n+i)}}(\sigma 0^\infty)$ as in Lemma 4.6 by using M_f and M' and store the result.
3. For each $m \in [p(n+i), p(n+i+1)]$ do the following:
 - a. Compute $A_n^{f_m}(\sigma 0^\infty)$ as in Lemma 4.6 by using M_f and M' and store the result.
 - b. Check if $|A_n^{f_m}(\sigma 0^\infty) - A_n^{f_{p(n+i)}}(\sigma 0^\infty)| \geq \frac{1}{2^{n+i+1}}$. If so, output 1.
4. Output 0.

It can be easily verified that N is a PSPACE machine. $r(n+i) = q(p(n+i+1)) + cn$ is a controlling polynomial for $\langle U_{n,i} \rangle_{n,i \in \mathbb{N}}$. Define

$$V_m = \bigcup_{\substack{n,i \geq 0 \\ n+i=m}} U_{n,i}.$$

Note that

$$\mu(V_m) \leq \frac{m}{2^m}.$$

It can be shown that for any j ,

$$\sum_{n>j} \frac{m}{2^m} = \frac{1}{2^{j-1}} + \frac{j}{2^j}.$$

Given any $k \geq 0$, let $p(k) = 3(k+1)$. Hence, we have

$$\sum_{n=p(k)+1}^{\infty} \frac{m}{2^m} = \frac{1}{2^{3(k+1)}} + \frac{3(k+1)}{2^{3(k+1)}} < \frac{1}{2^{k+1}} + \frac{1}{2^{k+1}} \frac{3(k+1)}{2^{2(k+1)}} < \frac{2}{2^{k+1}} = \frac{1}{2^k}.$$

The last inequality holds since $3(k+1) < 2^{2(k+1)}$ for all $k \geq 0$. Since each V_m is a finite union of sets from $\langle U_{n,i} \rangle_{n,i \in \mathbb{N}}$, the machine computing $\langle U_{n,i} \rangle_{n,i \in \mathbb{N}}$ can be easily modified to construct a machine witnessing that $\langle V_m \rangle_{m=1}^{\infty}$ is a PSPACE approximable sequence of sets. From these observations, it follows that $\langle V_m \rangle_{m=1}^{\infty}$ is a PSPACE Solovay test. Let x be a PSPACE random. x is in at most finitely many V_m and hence in at most finitely many $U_{n,i}$. Hence for some large enough N for all $n \geq N$, $i \geq 0$ and for all m such that $p(n+i) \leq m \leq p(n+i+1)$, we have $|A_n^{f_m}(x) - A_n^{f_{p(n+i)}}(x)| < \frac{1}{2^{n+i+1}}$. It follows that for all $n \geq N$ and for all $m \geq p(n)$ that

$$|A_n^{f_m}(x) - g_n(x)| = |A_n^{f_m}(x) - A_n^{f_{p(n)}}(x)| \leq \sum_{i=0}^{\infty} \frac{1}{2^{n+i+1}} \leq 2^{-n}.$$

Therefore, $\lim_{n \rightarrow \infty} \tilde{A}_n^f(x) = \lim_{n \rightarrow \infty} g_n(x)$ on all PSPACE random x and hence on all x which is EXP random.

Hence, we have shown that $\lim_{n \rightarrow \infty} \tilde{A}_n^f = \int f d\mu$ on EXP randoms which completes the proof of the theorem. \blacktriangleleft

6 General PSPACE ergodic theorem

We now extend Theorem 5.4 into the setting of PSPACE-probability Cantor spaces. V'yugin [26] shows that the speed of a.e. convergence to ergodic averages in computable ergodic systems is not computable in general. This leads us to consider some assumption on the rapidity of convergence in resource-bounded settings. We show that the requirement

on L^1 rapidity of convergence of A_n^f is sufficient to derive our result. Several probabilistic laws like the Law of Large Numbers, Law of Iterated Logarithm satisfy this criterion, hence the assumption is sufficiently general. Moreover, Theorem 5.1 shows that in the canonical example of Bernoulli systems with the left-shift, every PSPACE L^1 function exhibits PSPACE rapidity of A_n^f , showing that the latter property is not artificial. We prove the general PSPACE ergodic theorem for transformations which satisfy PSPACE ergodicity.

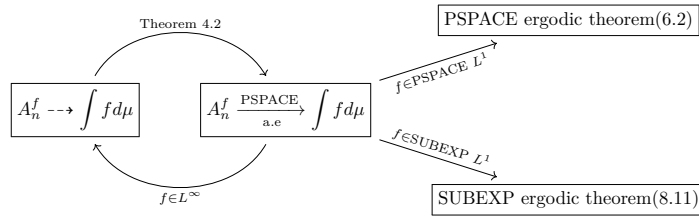
► **Definition 6.1** (PSPACE ergodic transformations). *A measurable function $T : (\Sigma^\infty, \mu) \rightarrow (\Sigma^\infty, \mu)$ is PSPACE ergodic if T is a PSPACE simple measure preserving transformation such that for any PSPACE L^1 -computable $f \in L^1(\Sigma^\infty, \mu)$, $\int f d\mu$ is a PSPACE-rapid L^1 limit point of A_n^f .*

Now, we prove the main result of our work.

► **Theorem 6.2.** *Let $(\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu)$ be a PSPACE-probability Cantor space. Let $T : (\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu) \rightarrow (\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu)$ be a PSPACE ergodic measure preserving transformation. Then, for any PSPACE L^1 -computable f , $\lim_{n \rightarrow \infty} \widetilde{A}_n^f = \int f d\mu$ on EXP randoms.*

Proof. Observe that Lemma 4.3, Corollary 4.4 and Lemma 4.6 are true in the setting of PSPACE-probability Cantor spaces. The proof of Lemma 4.5 can be extended to the setting of PSPACE-probability Cantor spaces in a straightforward manner. Since T is PSPACE ergodic, we get that $\int f d\mu$ is a PSPACE-rapid L^1 -limit point of A_n^f . Now, the theorem follows from these observations and the same techniques as in the proof of Theorem 5.4. ◀

The convergence notions involved in proving the PSPACE/SUBEXP-space ergodic theorems and their interrelationships are summarized in Figure 1.



■ **Figure 1** Relationships between the major convergence notions involving PSPACE simple measure preserving transformations. $A_n^f \dashrightarrow \int f d\mu$ denotes that $\int f d\mu$ is a PSPACE-rapid L^1 -limit point of A_n^f . PSPACE/SUBEXP-space ergodicity is required only for obtaining the ergodic theorems from PSPACE a.e convergence.

7 A partial converse to the PSPACE Ergodic Theorem

In this section we give a partial converse to the PSPACE ergodic theorem (Theorem 6.2). We show that for any PSPACE null x , there exists a function f and transformation T satisfying all the conditions in Theorem 6.2 such that $\widetilde{A}_n^f(x)$ does not converge to $\int f d\mu$.

Let us first observe that due to Corollary 4.4, Theorem 6.2 is equivalent to the following:

► **Theorem.** *Let T be a PSPACE ergodic measure preserving transformation such that for any PSPACE L^1 -computable f , $\int f d\mu$ is an PSPACE-rapid L^1 -limit point of A_n^f . Let $\{g_{n,i}\}$ be any collection of simple functions such that for each n , $\langle g_{n,i} \rangle_{i=1}^\infty$ is a PSPACE L^1 -approximation of \widetilde{A}_n^f . Then, $\lim_{n \rightarrow \infty} \lim_{i \rightarrow \infty} g_{n,i}(x) = \int f d\mu$ for any EXP random x .*

80:12 Ergodic Theorems and Converses for PSPACE Functions

Hence, the ideal converse to Theorem 6.2 is the following:

► **Theorem.** *Given any EXP null x , there exists a PSPACE ergodic measure preserving transformation T and PSPACE L^1 -computable $f \in L^1(\Sigma^\infty, \mu)$ such that the following conditions are true:*

1. $\int f d\mu$ is an PSPACE-rapid limit point of A_n^f .
2. There exists a collection of simple functions $\{g_{n,i}\}$ such that for each n , $\langle g_{n,i} \rangle_{i=1}^\infty$ is a PSPACE L^1 -approximation of A_n^f but $\lim_{n \rightarrow \infty} \lim_{i \rightarrow \infty} g_{n,i}(x) \neq \int f d\mu$.

But, we show the following partial converse to Theorem 6.2.

► **Theorem 7.1.** *Given any PSPACE null x , there exists a PSPACE L^1 -computable $f \in L^1(\Sigma^\infty, \mu)$ such that for any PSPACE simple measure preserving transformation, the following conditions are true:*

1. For all $n \in \mathbb{N}$, $\|A_n^f - \int f d\mu\|_1 = 0$. Hence, $\int f d\mu$ is an PSPACE-rapid L^1 -limit point of A_n^f .
2. There exists a collection of simple functions $\{g_{n,i}\}$ such that for each n , $\langle g_{n,i} \rangle_{i=1}^\infty$ is a PSPACE L^1 -approximation of A_n^f but $\lim_{n \rightarrow \infty} \lim_{i \rightarrow \infty} g_{n,i}(x) \neq \int f d\mu$.

A proof of the above theorem requires the construction in the following lemma.

► **Lemma 7.2.** *Let $\langle U_n \rangle_{n=1}^\infty$ be a PSPACE test. Then there exists a sequences of sets $\langle \widehat{S}_n \rangle_{n=1}^\infty$ such that for each $n \in \mathbb{N}$, $\widehat{S}_n \subseteq \Sigma^*$ satisfying the following conditions:*

1. $\mu([\widehat{S}_n]) \leq 2^{-n}$.
2. $\bigcap_{m=1}^\infty \bigcup_{n=m}^\infty [\widehat{S}_n] \supseteq \bigcap_{n=1}^\infty U_n$.
3. There exists $c \in \mathbb{N}$ such that for all n , $\sigma \in \widehat{S}_n$ implies $|\sigma| \leq n^c$.
4. There exists a PSPACE machine N such that $N(\sigma, 1^n) = 1$ if $\sigma \in \widehat{S}_n$ and 0 otherwise.

Proof of Theorem 7.1. Let $\langle V_n \rangle_{n=1}^\infty$ be any PSPACE test such that $x \in \bigcap_{n=1}^\infty V_n$. From Lemma 7.2, there exists a collection of sets $\langle \widehat{S}_n \rangle_{n=1}^\infty$ such that $\bigcap_{m=1}^\infty \bigcup_{n=m}^\infty [\widehat{S}_n] \supseteq \bigcap_{n=1}^\infty V_n$. Let,

$$U_n = \{\sigma : [\sigma] \in \widehat{S}_i \text{ for some } i \text{ such that } 2n+1 \leq i \leq 2(n+1)+1\}$$

Let $f_n = n\chi_{U_n}$. Since

$$\mu(U_n) \leq \sum_{i=2n+1}^{2(n+1)+1} \frac{1}{2^i} \leq \frac{1}{2^{2n}},$$

it follows that

$$\|f_n\|_1 \leq \frac{n}{2^{2n}} \leq \frac{1}{2^n}.$$

Using the properties of $\langle \widehat{S}_n \rangle_{n=1}^\infty$, it can be shown that $\langle f_n \rangle_{n=1}^\infty$ is a PSPACE L^1 -approximation of $f = 0$. We construct a machine M computing $\langle f_n \rangle_{n=1}^\infty$. The other conditions are easily verified. Let N be the machine from Lemma 7.2. On input $(1^n, \sigma)$, M does the following:

1. If $|\sigma| < (2(n+1)+1)^c$ then, output ?.
2. Else, for each $i \in [2n+1, 2(n+1)+1]$ do the following:
 - a. For each $\alpha \subseteq \sigma$, do the following:
 - i. If $N(1^i, \alpha) = 1$ then, output n .
3. Output 0.

M uses at most polynomial space and computes $\langle f_n \rangle_{n=1}^\infty$. Define

$$g_{n,i} = \frac{f_i + f_i \circ T + \cdots + f_i \circ T^{n-1}}{n}$$

For any fixed $n \in \mathbb{N}$, since T is a PSPACE simple transformation, as in Lemma 4.6 it can be shown that $\langle g_{n,i} \rangle_{i=1}^\infty$ is a PSPACE L^1 -approximation of A_n^f . We know that there exist infinitely many m such that $x \in \widehat{S}_m$. For any such m , let i be the unique number such that $2i + 1 \leq m \leq 2(i + 1) + 1$. For this i , $f_i(x) = i$. This shows that there exist infinitely many i such that $f_i(x) = i$. Since each f_i is a non-negative function, it follows that there are infinitely many i with $g_{n,i} \geq i/n$. Hence, if $\lim_{i \rightarrow \infty} g_{n,i}(x)$ exists, then it is equal to ∞ . It may be the case that $\lim_{i \rightarrow \infty} g_{n,i}(x)$ does not exist. In either case, $\lim_{n \rightarrow \infty} \lim_{i \rightarrow \infty} g_{n,i}(x)$ cannot be equal to $\int f d\mu = 0$. Hence, our construction satisfies all the desired conditions. \blacktriangleleft

8 An ergodic theorem for SUBEXP-space randoms and its converse

In the previous sections, we demonstrated that for PSPACE L^1 -computable functions and PSPACE simple transformations, the Birkhoff averages converge to the desired value over EXP randoms. However, the converse holds only over PSPACE non-randoms. The two major reasons for this *gap* are the following: PSPACE-rapid convergence necessitates exponential length cylinders while constructing the randomness tests, and PSPACE L^1 -computable functions are not strong enough to *capture* all PSPACE randoms. In this section, we demonstrate that for a different notion of randomness - SUBEXP-space randoms and a larger class of L^1 -computable functions (SUBEXP-space L^1 -computable), we can prove the ergodic theorem on the randoms and obtain its converse on the non-randoms. Analogous to Towsner and Franklin [5], we demonstrate that the ergodic theorem for PSPACE simple transformations and SUBEXP-space L^1 -computable functions satisfying PSPACE rapidity, fails for exactly this class of non-random points. We first introduce SUBEXP-space tests and SUBEXP-space randomness.

► **Definition 8.1** (SUBEXP-space sequence of open sets). *A sequence of open sets $\langle U_n \rangle_{n=1}^\infty$ is a SUBEXP-space sequence of open sets if there exists a sequence of sets $\langle S_n^k \rangle_{k,n \in \mathbb{N}}$, where $S_n^k \subseteq \Sigma^*$ such that*

1. $U_n = \bigcup_{k=1}^\infty [S_n^k]$, where for any $m > 0$, $\mu(U_n - \bigcup_{k=1}^m [S_n^k]) \leq m^{-\log(m)}$.
2. There exists a controlling polynomial p such that $\max\{|\sigma| : \sigma \in \bigcup_{k=1}^m S_n^k\} \leq 2^{p(\log(n) + \log(m))}$.
3. The function $g : \Sigma^* \times 1^* \times 1^* \rightarrow \{0, 1\}$ such that $g(\sigma, 1^n, 1^m) = 1$ if $\sigma \in S_n^m$, and 0 otherwise, is decidable by a PSPACE machine.

► **Definition 8.2** (SUBEXP-space randomness). *A sequence of open sets $\langle U_n \rangle_{n=1}^\infty$ is a SUBEXP-space test if it is a SUBEXP-space sequence of open sets and for all $n \in \mathbb{N}$, $\mu(U_n) \leq n^{-\log(n)}$.*

*A set $A \subseteq \Sigma^\infty$ is SUBEXP-space null if there is a SUBEXP-space test $\langle U_n \rangle_{n=1}^\infty$ such that $A \subseteq \bigcap_{n=1}^\infty U_n$ and is SUBEXP-space random otherwise.*⁵

The slower decay rate of $n^{-\log(n)} = 2^{-\log(n)^2}$ enables us to obtain an ergodic theorem and an exact converse in the SUBEXP-space setting. The following result is useful in manipulating sums involving terms of the form $2^{-(\log(n))^k}$ for $k \geq 2$.

⁵ It is easy to see that the set of SUBEXP-space randoms is smaller than the set of PSPACE randoms. But, we do not know if any inclusion holds between SUBEXP-space randoms and EXP-randoms.

80:14 Ergodic Theorems and Converses for PSPACE Functions

► **Lemma 8.3.** For any $m \in \mathbb{N}$, $\sum_{n=2(2m^2+1)}^{\infty} \frac{n}{n^{\log(n)}} \leq \frac{1}{m^{\log(m)}}$.

A similar inequality holds on replacing $n/n^{\log(n)}$ with $1/n^{\log(n)}$. Now, we introduce the Solovay analogue of SUBEXP-space randomness and prove that these notions are analogous.

► **Definition 8.4** (SUBEXP-space Solovay test). A sequence of open sets $\langle U_n \rangle_{n=1}^{\infty}$ is a SUBEXP-space Solovay test if it is a SUBEXP-space sequence of open sets and there exists a polynomial p such that $\forall m \geq 0$, $\sum_{n=p(m)+1}^{\infty} \mu(U_n) \leq m^{-\log(m)}$. A set $A \subseteq \Sigma^{\infty}$ is SUBEXP-space Solovay null if there exists a SUBEXP-space Solovay test $\langle U_n \rangle_{n=1}^{\infty}$ such that $A \subseteq \bigcap_{i=1}^{\infty} \bigcup_{n=i}^{\infty} U_n$, and is SUBEXP-space Solovay random otherwise.

► **Lemma 8.5.** A set $A \subseteq \Sigma^{\infty}$ is SUBEXP-space null if and only if A is SUBEXP-space Solovay null.

Now, we define SUBEXP-space analogues of concepts from Section 3.

► **Definition 8.6** (SUBEXP-space sequence of simple functions). A sequence of simple functions $\langle f_n \rangle_{n=1}^{\infty}$ where each $f_n : \Sigma^{\infty} \rightarrow \mathbb{Q}$ is a SUBEXP-space sequence of simple functions if

1. There is a controlling polynomial p such that for each n , there exists $k(n) \in \mathbb{N}$, $\{d_1, d_2, \dots, d_{k(n)}\} \subseteq \mathbb{Q}$ and $\{\sigma_1, \sigma_2, \dots, \sigma_{k(n)}\} \subseteq \Sigma^{2^{p(\log(n))}}$ such that $f_n = \sum_{i=1}^{k(n)} d_i \chi_{\sigma_i}$, where χ_{σ_i} is the characteristic function of the cylinder $[\sigma_i]$.
2. There is a PSPACE machine M such that for each $n \in \mathbb{N}$, $\sigma \in \Sigma^*$, $M(1^n, \sigma)$ outputs $f_n(\sigma 0^{\infty})$ if $|\sigma| \geq 2^{p(\log(n))}$ and ? otherwise.

► **Definition 8.7** (SUBEXP-space L^1 -computable functions). A function $f \in L^1(\Sigma^{\infty}, \mu)$ is SUBEXP-space L^1 -computable if there exists a SUBEXP-space sequence of simple functions $\langle f_n \rangle_{n=1}^{\infty}$ such that for every $n \in \mathbb{N}$, $\|f - f_n\| \leq n^{-\log(n)}$. The sequence $\langle f_n \rangle_{n=1}^{\infty}$ is called a SUBEXP-space L^1 -approximation of f .

We require the following equivalent definitions of PSPACE-rapid convergence notions for working in the setting of SUBEXP-space randomness.

► **Lemma 8.8.** A real number a is a PSPACE-rapid limit point of the real number sequence $\langle a_n \rangle_{n=1}^{\infty}$ if and only if there exists a polynomial p such that for all $m \in \mathbb{N}$, $\exists k \leq 2^{p(\log(m))}$ such that $|a_k - a| \leq m^{-\log(m)}$.

► **Lemma 8.9.** A sequence of measurable functions $\langle f_n \rangle_{n=1}^{\infty}$ is PSPACE-rapid almost everywhere convergent to a measurable function f if and only if there exists a polynomial p such that for all m_1 and m_2 ,

$$\mu \left(\left\{ x : \sup_{n \geq 2^{p(\log(m_1) + \log(m_2))}} |f_n(x) - f(x)| \geq \frac{1}{m_1^{\log(m_1)}} \right\} \right) \leq \frac{1}{m_2^{\log(m_2)}}.$$

The same technique used in the proof of Lemma 8.8 can be used to prove this claim.

Before addressing the main result, let us define SUBEXP-space ergodicity.

► **Definition 8.10** (SUBEXP-space ergodic transformations). A measurable function $T : (\Sigma^{\infty}, \mu) \rightarrow (\Sigma^{\infty}, \mu)$ is SUBEXP-space ergodic if T is a PSPACE simple transformation such that for any SUBEXP-space L^1 -computable $f \in L^1(\Sigma^{\infty}, \mu)$, $\int f d\mu$ is a PSPACE-rapid L^1 limit point of $A_n^{f, T}$.

Lemma 4.3, Corollary 4.4 and Lemma 4.6 have analogous results in the SUBEXP-space setting. We prove the SUBEXP-space ergodic theorem below.

► **Theorem 8.11.** *Let $T : (\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu) \rightarrow (\Sigma^\infty, \mathcal{B}(\Sigma^\infty), \mu)$ be a SUBEXP-space ergodic measure preserving transformation. Then, for any SUBEXP-space L^1 -computable f , $\lim_{n \rightarrow \infty} \widetilde{A}_n^f = \int f d\mu$ on SUBEXP-space randoms.*

An important reason for investigating SUBEXP-space randomness is that the SUBEXP-space ergodic theorem has an exact converse unlike the PSPACE ergodic theorem which only seems to have a partial converse (Theorem 7.1).

► **Theorem 8.12.** *Given any SUBEXP-space null x , there exists a SUBEXP-space L^1 -computable $f \in L^1(\Sigma^\infty, \mu)$ such that for any PSPACE simple measure preserving transformation, the following conditions are true:*

1. *For all $n \in \mathbb{N}$, $\|A_n^f - \int f d\mu\|_1 = 0$. Hence, $\int f d\mu$ is an PSPACE-rapid L^1 -limit point of A_n^f .*
2. *There exists a collection of simple functions $\{g_{n,i}\}$ such that for each n , $\langle g_{n,i} \rangle_{i=1}^\infty$ is a SUBEXP-space L^1 -approximation of A_n^f but $\lim_{n \rightarrow \infty} \lim_{i \rightarrow \infty} g_{n,i}(x) \neq \int f d\mu$.*

The proofs of both Theorem 8.11 and Theorem 8.12 are similar to those of Theorem 6.2 and Theorem 7.1, but requires Lemma 8.3 for minimizing summations of the form $\sum n^{-\log(n)}$ appearing in the error bounds.

References

- 1 L. Bienvenu, A. Day, M. Hoyrup, I. Mezhirov, and A. Shen. A constructive version of Birkhoff's ergodic theorem for Martin-Löf random points. *Information and Computation*, 210:21–30, 2012.
- 2 P. Billingsley. *Probability and Measure*, second edition. John Wiley and Sons, New York, N.Y., 1986.
- 3 E. Bishop. *Foundations of constructive analysis*. McGraw-Hill, New York, 1967.
- 4 Rodney G Downey and Denis R Hirschfeldt. *Algorithmic randomness and complexity*. Springer Science & Business Media, 2010.
- 5 Johanna NY Franklin and Henry Towsner. Randomness and non-ergodic systems. *Moscow Mathematical Journal*, 14:711–744, 2014.
- 6 Stefano Galatolo, Mathieu Hoyrup, and Cristóbal Rojas. A constructive Borel–Cantelli lemma. constructing orbits with required statistical properties. *Theoretical Computer Science*, 410(21-23):2207–2222, 2009.
- 7 S. Galatolo, M. Hoyrup, and C. Rojas. Computing the speed of convergence of ergodic averages and pseudorandom points in computable dynamical systems. In *7th International Conference on Computability and Complexity in Analysis*, pages 7–18, 2010.
- 8 Michael Hochman. Upcrossing inequalities for stationary sequences and applications. *Annals of Probability*, 37(6):2135–2149, 2009.
- 9 Mathieu Hoyrup. The dimension of ergodic random sequences. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, pages 567–576, 2012.
- 10 Mathieu Hoyrup and Cristóbal Rojas. Applications of effective probability theory to Martin-Löf randomness. In *International Colloquium on Automata, Languages, and Programming*, pages 549–561. Springer, 2009.
- 11 Mathieu Hoyrup and Cristóbal Rojas. Computability of probability measures and martin-löf randomness over metric spaces. *Information and Computation*, 207(7):830–847, 2009.
- 12 Xiang Huang and D. M. Stull. Polynomial space randomness in analysis. In *MFCS*, 2016.
- 13 Ker-I Ko. *Complexity theory of real functions*. Springer Science & Business Media, 2012.
- 14 Ulrich Krengel. On the speed of convergence in the ergodic theorem. *Monatshefte für Mathematik*, 86:3–6, 1978.
- 15 L. Kuipers and H. Niederreiter. *Uniform distribution of sequences*. Wiley-Interscience [John Wiley & Sons], New York-London-Sydney, 1974. Pure and Applied Mathematics.
- 16 John E. Maxfield. A short proof of Pillai's theorem on normal numbers. *Pacific Journal of Mathematics*, 2(1):23–24, 1952.

- 17 S. Nandakumar. An effective ergodic theorem and some applications. In *Proceedings of the 40th Annual Symposium on the Theory of Computing*, pages 39–44, 2008.
- 18 Satyadev Nandakumar, Subin Pulari, Prateek Vishnoi, and Gopal Viswanathan. An analogue of pillai’s theorem for continued fraction normality and an application to subsequences. *arXiv preprint*, 2019. [arXiv:1909.03431](https://arxiv.org/abs/1909.03431).
- 19 André Nies. *Computability and randomness*, volume 51. OUP Oxford, 2009.
- 20 S. S. Pillai. On normal numbers. *Proceedings of the Indian Academy of Sciences Sect. A*, 11:73–80, 1940.
- 21 Jason Rute. *Topics in algorithmic randomness and computable analysis*. PhD thesis, Carnegie Mellon University, August 2013.
- 22 Donald M Stull. *Algorithmic randomness and analysis*. PhD thesis, Iowa State University, Ames, Iowa, USA, 2017.
- 23 Donald M. Stull. Resource bounded randomness and its applications. *Algorithmic Randomness: Progress and Prospects*, 50:301, 2020.
- 24 M. van Lambalgen. *Random Sequences*. PhD thesis, Department of Mathematics, University of Amsterdam, 1987.
- 25 V. G. Vovk. The law of the iterated logarithm for random Kolmogorov, or chaotic, sequences. *Theory of Probability and its Applications*, 32(3):413–425, 1988.
- 26 V. V. V’yugin. Effective convergence in probability and an ergodic theorem for individual random sequences. *Theory of Probability and Its Applications*, 42(1):39–50, 1997.

A Proof of Theorem 5.1

Proof of Theorem 5.1. Let $\langle f_n \rangle_{n=1}^\infty$ be a PSPACE sequence of simple functions witnessing the fact that f is PSPACE L^1 -computable. Let p be a controlling polynomial and let t be a polynomial upper bound for the space complexity of the machine associated with $\langle f_n \rangle_{n=1}^\infty$. Let q_1, q_2 be the polynomials from Lemma 5.2. Let $c \in \mathbb{N}$ be any number such that if a $r \in \mathbb{Q}$ has a representation of length l then $r \leq 2^{l^c}$ (see Section 2). Observe that for any $m \in \mathbb{N}$,

$$\begin{aligned} \|A_n^f - \int f d\mu\|_1 &\leq \|A_n^f - A_n^{f_{q_1(m+3)}}\|_1 + \|A_n^{f_{q_1(m+3)}} - \int f_{q_1(m+3)} d\mu\|_1 + \left\| \int f_{q_1(m+3)} d\mu - \int f d\mu \right\|_1 \\ &\leq \frac{1}{2^{q_1(m+3)}} + \|A_n^{f_{q_1(m+3)}} - \int f_{q_1(m+3)} d\mu\|_1 + \frac{1}{2^{q_1(m+3)}}. \\ &\leq \frac{1}{2^{m+3}} + \|A_n^{f_{q_1(m+3)}} - \int f_{q_1(m+3)} d\mu\|_1 + \frac{1}{2^{m+3}}. \end{aligned}$$

We know that there exist $\{\sigma_1, \sigma_2 \dots \sigma_k\} \subseteq \Sigma^{p(q_1(m+3))}$ such that $A_n^{f_{q_1(m+3)}} = \sum_{i=1}^{k(q_1(m+3))} d_i \chi_{\sigma_i}$ where each $d_i \leq 2^{t(q_1(m+3)+p(q_1(m+3)))^c}$. Hence,

$$\|A_n^{f_{q_1(m+3)}} - \int f_{q_1(m+3)} d\mu\|_1 \leq 2^{t(q_1(m+3)+p(q_1(m+3)))^c} \sum_{i=1}^{k(q_1(m+3))} \|A_n^{\chi_{\sigma_i}} - \mu(\sigma_i)\|_1$$

Since $|\sigma_i| \geq p(q_1(m+3)) \geq q_1(m+3)$, using Lemma 5.2, for

$$n \geq p(q_1(m+3))^3 2^{q_2(t(q_1(m+3)+p(q_1(m+3)))^c + p(q_1(m+3)) + m + 3)}$$

we get that,

$$\begin{aligned} \|A_n^{f_{q_1(m+3)}} - \int f_{q_1(m+3)} d\mu\|_1 &\leq \frac{2^{t(q_1(m+3)+p(q_1(m+3)))^c + p(q_1(m+3))}}{2^{t(q_1(m+3)+p(q_1(m+3)))^c + p(q_1(m+3)) + m + 3}} \\ &\leq \frac{1}{2^{m+3}}. \end{aligned}$$

Hence, for all $n \geq p(q_1(m+3))^3 2^{q_2(t(q_1(m+3)+p(q_1(m+3)))^c + p(q_1(m+3)) + m + 3)}$ we have $\|A_n^f - \int f d\mu\|_1 \leq 3 \cdot 2^{-(m+3)} < 2^{-m}$. \blacktriangleleft

Proof of Lemma 5.2. The major difficulty in directly approximating $\|A_n^{X_\sigma} - \mu(\sigma)\|_1$ is that for any $n, m \in \mathbb{N}$, $A_n^{X_\sigma}$ and $A_m^{X_\sigma}$ may not be *independent*. In order to overcome this, we use constructions similar to those used in proving Pillai's theorem (see [20], [16] for normal numbers, [18] for continued fractions) in order to approximate each $A_n^{X_\sigma}$ with sums of *disjoint* averages. These *disjoint* averages turns out to be averages of independent random variables. Hence, elementary results from probability theory regarding independent random variables can be used to show that $A_n^{X_\sigma}$ converges to $\int f d\mu$ sufficiently fast.

Observe that for any $x \in \Sigma^\infty$

$$A_n^{X_\sigma}(x) = \frac{|\{i \in [0, n-1] \mid T^i x \in [\sigma]\}|}{n}$$

Let $k = |\sigma|$. As in the proof of Theorem 3.1 from [18], the following is a decomposition of the above term as *disjoint* averages,

$$\frac{|\{i \in [0, n-1] \mid T^i x \in [\sigma]\}|}{n} = g_1(n) + g_2(n) + \cdots + g_{(1 + \lfloor \log_2 \frac{n}{k} \rfloor)}(n) + \frac{(k-1) \cdot O(\log n)}{n}$$

where

$$g_p(n) = \begin{cases} \frac{|\{i \mid T^{ki} x \in [\sigma], 0 \leq i \leq \lfloor n/k \rfloor\}|}{n} & \text{if } p = 1 \\ \frac{\sum_{j=1}^{k-1} |\{i \mid T^{(2^{p-1})ki} x \in [S_j], 0 \leq i \leq \lfloor n/2^{p-1}k \rfloor\}|}{n} & \text{if } 1 < p \leq (1 + \lfloor \log_2(n/k) \rfloor) \\ 0 & \text{otherwise,} \end{cases}$$

and S_j is the finite collection of $2^{(p-1)}k$ length blocks s.t σ occurs in it at starting position $(2^{(p-2)}k - j + 1)^{th}$ position i.e S_j is the set of strings of the form, $u a_1 a_2 \dots a_k v$ where u is some string of length $2^{p-2}k - j$, and v is some string of length $2^{p-2}k - k + j$.

When $p = 1$,

$$g_1(n) = \frac{\sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} X_i^{1,1}}{n}$$

where

$$X_i^{1,1}(x) = \begin{cases} 1 & \text{if } x[ik+1, (i+1)k] = \sigma \\ 0 & \text{otherwise} \end{cases}$$

When $1 < p \leq \lfloor \log_2(n/k) \rfloor$,

$$g_p(n) = \frac{\sum_{i=1}^{\lfloor \frac{n}{2^{p-1}k} \rfloor} \sum_{j=1}^{k-1} X_i^{p,j}}{n}$$

where,

$$X_i^{p,j}(x) = \begin{cases} 1 & \text{if } x[2^{p-2}k - j + 1, 2^{p-2}k - j + k] = \sigma \\ 0 & \text{otherwise} \end{cases}$$

Hence,

$$A_n^{X_\sigma}(x) = \frac{\sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} X_i^{1,1}(x)}{n} + \sum_{p=2}^{\lfloor \log_2(\frac{n}{k}) \rfloor} \sum_{j=1}^{k-1} \frac{\sum_{i=1}^{\lfloor \frac{n}{2^{p-1}k} \rfloor} X_i^{p,j}}{n} + \frac{(k-1) \cdot O(\log n)}{n}$$

80:18 Ergodic Theorems and Converses for PSPACE Functions

An important observation that we use later in the proof is that for any fixed p and j , $\{X_i^{p,j}\}_{i=1}^{\infty}$ is a collection of i.i.d Bernoulli random variables such that $\mu(\{x : X_i^{p,j}(x) = 1\}) = 2^{-|s|}$. We will show that the conclusion of the lemma holds when $q_1(m) = 2(m+6)$ and $q_2(m) = 5(m+6)$. For any $m \in \mathbb{N}$,

$$\left\| \sum_{p=m+5+2}^{\infty} \sum_{j=1}^{k-1} \frac{1}{n} \sum_{i=1}^{\lfloor \frac{n}{2^{p-1}k} \rfloor} X_i^{p,j} \right\|_2 \leq \sum_{p=m+5+2}^{\infty} \frac{1}{2^{p-1}} \leq \frac{1}{2^{m+5}} \quad (1)$$

And for $n \geq |\sigma|^{32} 2^{2(m+5)} > |\sigma|^2 2^{2(m+5)}$,

$$\left\| \frac{(k-1)O(\log(n))}{n} \right\|_2 = \left\| \frac{(k-1)O(\log(n))}{\sqrt{n}\sqrt{n}} \right\|_2 \leq \left| \frac{k-1}{\sqrt{n}} \right| \leq \left| \frac{k-1}{k2^{m+5}} \right| \leq \frac{1}{2^{m+5}} \quad (2)$$

Let,

$$D_{n,m}^{\sigma}(x) = \frac{\sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} X_i^{1,1}(x)}{n} + \sum_{p=2}^{m+5+2} \sum_{j=1}^{k-1} \frac{\sum_{i=1}^{\lfloor \frac{n}{2^{p-1}k} \rfloor} X_i^{p,j}}{n}$$

From (1) and (2), we get that

$$\|A_n^{\chi\sigma} - D_{n,m}^{\sigma}\|_2 \leq \frac{2}{2^{m+5}}.$$

Let,

$$E_{n,m}^{\sigma}(x) = \left(\frac{\sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} X_i^{1,1}(x)}{\lfloor \frac{n}{k} \rfloor} - \frac{1}{2^k} \right) \frac{\lfloor \frac{n}{k} \rfloor}{n} + \sum_{p=2}^{m+5+2} \sum_{j=1}^{k-1} \left(\frac{\sum_{i=1}^{\lfloor \frac{n}{2^{p-1}k} \rfloor} X_i^{p,j}}{\lfloor \frac{n}{2^{p-1}k} \rfloor} - \frac{1}{2^k} \right) \frac{\lfloor \frac{n}{2^{p-1}k} \rfloor}{n}$$

Now,

$$D_{n,m}^{\sigma}(x) - E_{n,m}^{\sigma}(x) = \frac{1}{2^k k} + \sum_{p=2}^{m+5+2} \sum_{j=1}^{k-1} \frac{1}{2^k} \frac{\lfloor \frac{n}{2^{p-1}k} \rfloor}{n}$$

It follows that,

$$\begin{aligned} \|D_{n,m}^{\sigma}(x) - E_{n,m}^{\sigma}\|_2 &\leq \frac{1}{2^k} + \sum_{p=2}^{m+5+2} \sum_{j=1}^{k-1} \frac{1}{2^k 2^{p-1}k} \\ &\leq \frac{1}{2^k} + \sum_{p=2}^{m+5+2} \frac{1}{2^k 2^{p-1}} \\ &\leq \frac{1}{2^k} + \sum_{p=2}^{m+5+2} \frac{1}{2^k} \\ &\leq \frac{m+5+2}{2^k} \end{aligned}$$

Hence, if $|\sigma| = k \geq q_1(m) = m + 5 + 2 + m + 5$ then,

$$\|D_{n,m}^\sigma(x) - E_{n,m}^\sigma\|_2 \leq \frac{1}{2^{m+5}}$$

and,

$$\begin{aligned} \|A_n^{X_\sigma} - \mu(\sigma)\|_2 &\leq \|A_n^{X_\sigma} - D_{n,m}^\sigma\|_2 + \|D_{n,m}^\sigma(x) - E_{n,m}^\sigma\|_2 + \|E_{n,m}^\sigma\|_2 + \frac{1}{2^k} \\ &\leq \frac{3}{2^{m+5}} + \|E_{n,m}^\sigma\|_2 + \frac{1}{2^{2m+12}} \\ &\leq \frac{4}{2^{m+5}} + \|E_{n,m}^\sigma\|_2. \end{aligned}$$

Hence, in order to show that for all $n \geq |\sigma|^3 2^{q_2(m)}$, $\|A_n^{X_\sigma} - \mu(\sigma)\|_1 \leq \|A_n^{X_\sigma} - \mu(\sigma)\|_2 \leq 2^{-m}$, it is enough to show that for all $n \geq |\sigma|^3 2^{q_2(m)}$, $\|E_{n,m}^\sigma\|_2 \leq 2^{-(m+5)}$. Observe that

$$\|E_{n,m}^\sigma\|_2 \leq \left\| \frac{1}{\lfloor \frac{n}{k} \rfloor} \sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} X_i^{1,1}(x) - \frac{1}{2^k} \right\|_2 + \sum_{p=2}^{m+5+2} \sum_{j=1}^{k-1} \left\| \frac{1}{\lfloor \frac{n}{2^{p-1}k} \rfloor} \sum_{i=1}^{\lfloor \frac{n}{2^{p-1}k} \rfloor} X_i^{p,j} - \frac{1}{2^k} \right\|_2.$$

Let Y_1, Y_2, \dots, Y_n be i.i.d Bernoulli random variables,

$$\begin{aligned} \left\| \frac{1}{n} \sum_{i=1}^n Y_i - \mathbf{E}(Y_1) \right\|_2 &= \sqrt{\mathbf{E} \left(\left(\frac{1}{n} \sum_{i=1}^n Y_i - \mathbf{E}(Y_1) \right)^2 \right)} \\ &= \sqrt{\text{Var} \left(\frac{1}{n} \sum_{i=1}^n Y_i \right)} \\ &= \sqrt{\frac{1}{n^2} n \text{Var}(Y_1)} \\ &\leq \frac{\sqrt{\text{Var}(Y_1)}}{\sqrt{n}} \\ &\leq \frac{1}{2\sqrt{n}} \end{aligned}$$

The last inequality follows from the fact that the variance of Bernoulli random variables are always bounded by $\frac{1}{4}$. Hence, if $n \geq |\sigma|^3 2^{q_2(m)} = |\sigma|^3 2^{5(m+6)}$ then,

$$\left\lfloor \frac{n}{k} \right\rfloor > k^2 2^{4(m+6)}$$

and

$$\left\lfloor \frac{n}{2^{p-1}k} \right\rfloor \geq \frac{k^3 2^{5(m+6)}}{2^{m+5+1}k} > k^2 2^{4(m+6)}.$$

Hence for all $n \geq |\sigma|^3 2^{q_2(m)} = |\sigma|^3 2^{5(m+6)}$,


$$\begin{aligned} \|E_{n,m}^\sigma\|_2 &\leq \frac{1}{2k2^{2(m+6)}} + (m+6)k \frac{1}{2k2^{2(m+6)}} \\ &< \frac{1}{2^{m+6}} + \frac{1}{2^{m+6}} \\ &\leq \frac{1}{2^{m+5}}. \end{aligned}$$

Hence we obtain the desired conclusion. ◀

On Guidable Index of Tree Automata

Damian Niwiński  

Institute of Informatics, University of Warsaw, Poland

Michał Skrzypczak  

Institute of Informatics, University of Warsaw, Poland

Abstract

We study guidable parity automata over infinite trees introduced by Colcombet and Löding, which form an expressively complete subclass of all non-deterministic tree automata. We show that, for any non-deterministic automaton, an equivalent guidable automaton with the smallest possible index can be effectively found. Moreover, if an input automaton is of a special kind, i.e. it is deterministic or game automaton then a guidable automaton with an optimal index can be deterministic (respectively game) automaton as well. Recall that the problem whether an equivalent non-deterministic automaton with the smallest possible index can be effectively found is open, and a positive answer is known only in the case when an input automaton is a deterministic, or more generally, a game automaton.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases guidable automata, index problem, ω -regular games

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.81

Funding *Michał Skrzypczak*: Supported by the Polish NCN grant 2016/22/E/ST6/00041.

1 Introduction

Rabin automata on infinite trees constitute one of the most expressive formalisms based on finite-state recognisability. Introduced by Rabin in the proof of his seminal decidability result [19] as an enhancement of the more basic concept of Büchi automata on ω -words [2], tree automata continue to attract attention of researchers, and make a crossing point of various areas like logic, games, set-theoretic topology, fixed-point calculi, and theory of verification. It is an intriguing fact that a number of questions, which are by now well understood for automata on ω -words, are still largely unsolved for automata on trees. This concerns in particular decidability questions like effective measurability and effective classification within various hierarchies based on topology, or on the structure of automata. While it is plausible that we reach the frontiers of decidability here, there is neither any evidence that these problems may not be decidable.

Among the hierarchies mentioned above, the Rabin-Mostowski index hierarchy has attracted a special attention, owing to its close relation to an alternation hierarchy in the μ -calculus (cf. [1, 17]), and to the complexity of the non-emptiness problem. It refers to the acceptance criterion, which for infinite computations is conveniently expressed in terms of colours associated with the automaton states, that may or should repeat infinitely often. In a general setting, the criterion specifies explicitly the requested combinations of colours (Muller criterion), which refines the original Büchi criterion that has just requested some “good” colour to repeat infinitely often. While the Büchi and Muller criteria are equivalent for non-deterministic automata on ω -words [2], they are not for automata on trees as noted already by Rabin [20]. The parity criterion is derived from Büchi’s idea by distinguishing between “good” and “bad” colours and requesting that the highest-ranked colour repeating infinitely often is good. This is implemented by representing good and bad colours by even and odd integers, respectively. An *index* is a set of colours $\{i, i+1, \dots, j\}$ used by an automaton; its size corresponds to the number of alternations between good and bad



© Damian Niwiński and Michał Skrzypczak;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 81; pp. 81:1–81:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

colours. The parity criterion is expressively equivalent to the Muller criterion, but it is more succinct and leads to the better complexity of the non-emptiness problem, up to our present knowledge. (See [21] for an insightful discussion of other meaningful criteria.)

The complexity of the non-emptiness problem for automata on infinite trees is actually a pertinent open question. While the problem is NP-complete for automata with the original Rabin acceptance criterion [9], it is expectedly more feasible for automata with the parity criterion, being in turn equivalent to the celebrated problem of solving parity games [10]. It is well-known that the complexity of this last problem has been recently improved [4] roughly from $n^{\mathcal{O}(d)}$ to $n^{\mathcal{O}(\log d)}$, where n is the number of positions and d the number of priorities (colours). This yields a quasi-polynomial solution for the non-emptiness problem for parity tree automata as well, with d corresponding to the size of automaton's index. Whether this improvement makes the quest for a small index more or less important may be the subject of discussion. In prospective applications in the model checking, the index d is derived from the formula whereas n corresponds to the size of the system, and is usually much bigger than d , so that even a small improvement of the index may be advantageous. But in any case, the recent development for parity games shows that the index is an essential parameter in understanding the problem.

The relevance of this parameter has been noticed already by Mostowski [15]. While it is known that in general an arbitrarily high index is needed for non-deterministic [16] as well as for alternating [1] automata, it is generally open if an automaton with an optimal index can be effectively found. Algorithms were given only in the special cases, where the input automaton is deterministic [18], or more generally it is a so-called game automaton [11].

An interesting approach has been undertaken by Colcombet and Löding [7], who reduced the non-deterministic index problem of parity tree automata to a problem of an apparently different nature, concerning the asymptotic behaviour of counters in some automata related to the (classical) star-height problem. In the course of their proof, these authors introduced an auxiliary concept of a *guidable tree automaton*, which in our opinion has a compelling potential and deserves to be better understood. Intuitively, a guidable automaton behaves almost like a deterministic automaton if it is given (as a kind of oracle) the non-deterministic choices of *any* other automaton running on the same tree. Colcombet and Löding [7] showed in particular that any automaton can be transformed into an equivalent guidable automaton, which essentially realises a strategy of Pathfinder in the celebrated game introduced by Gurevich and Harrington [13]. Although a guidable automaton is unsurprisingly not unique, it nevertheless constitutes a kind of a normal form of a non-deterministic automaton. It should be stressed that canonical forms are generally missing in the theory of automata on infinite objects, which is one of the sources of difficulties there. Therefore, we believe that the idea of guidable automata is worth to pursue.

In the present paper we show how to effectively compute an equivalent guidable automaton with the smallest possible index – among all guidable automata – without any restriction on the input automaton. We also revisit the concept of game automata mentioned above and show, relying on the construction of a guidable automaton by Colcombet and Löding [7], that any game automaton is itself guidable. Moreover, if an input automaton is a game or deterministic automaton then a guidable automaton with an optimal index can be a game (respectively, deterministic) automaton as well.

Unfortunately, the guidable index can be in general arbitrarily worse than the smallest possible index of an equivalent non-deterministic automaton, we are primarily searching for. We believe however, that the games used in our proofs may be of potential interest for the main problem. Note that similar games have been used in a recent work [5] (see [6]) to decide separability of regular tree languages by deterministic and game automata.

2 Basic notions

The set of natural numbers $\{0, 1, \dots\}$ is denoted ω . An *alphabet* is any finite non-empty set Σ of *letters*. By Σ^* we denote the set of *words* (i.e. finite sequences) over an alphabet Σ . By $u|_n$ we denote the finite sequence consisting of the first n symbols of u . The empty sequence is denoted ϵ . By $|u|$ we denote the length of a finite sequence u .

A Σ -labelled *tree* (shortly *tree*) is a function $t: \{\mathsf{L}, \mathsf{R}\}^* \rightarrow \Sigma$, where $\mathsf{L} \neq \mathsf{R}$ are two special symbols called *directions*. The set of all such trees is denoted Tr_Σ . If Σ and Γ are two alphabets then each pair of trees $t_1 \in \text{Tr}_\Sigma$ and $t_2 \in \text{Tr}_\Gamma$ induces their *product* $t_1 \otimes t_2 \in \text{Tr}_{\Sigma \times \Gamma}$, with $(t_1 \otimes t_2)(u) \stackrel{\text{def}}{=} (t_1(u), t_2(u))$. Given a tree language $L \subseteq \text{Tr}_{\Sigma \times \Gamma}$, its *projection* onto Σ is the set of trees $t_1 \in \text{Tr}_\Sigma$, such that there exists a tree $t_2 \in \text{Tr}_\Gamma$, for which $t_1 \otimes t_2 \in L$.

An infinite sequence of directions $b \in \{\mathsf{L}, \mathsf{R}\}^\omega$ is called a *branch*. Similarly, a *path* in a tree $t \in \text{Tr}_A$ is a sequence $(a_n, d_n)_{n \in \omega} \in (\Sigma \times \{\mathsf{L}, \mathsf{R}\})^\omega$, such that $a_n = t(d_0 \cdots d_{n-1})$ for $n = 0, 1, \dots$

An *index* is a non-empty finite range of natural numbers $C = \{i, i+1, \dots, j\} \subseteq \omega$. Elements $c \in C$ are called *priorities*. We say that an infinite sequence of priorities $(c_n)_{n \in \omega}$ is *parity accepting* if $\limsup_{n \rightarrow \infty} c_n \equiv 0 \pmod{2}$. Otherwise, we say that such a sequence is *parity rejecting*.

A *non-deterministic C-parity tree automaton* (shortly *non-deterministic automaton*) is a tuple $\mathcal{A} = \langle \Sigma, Q^{\mathcal{A}}, q_{\mathsf{I}}^{\mathcal{A}}, \Delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle$, where Σ is an alphabet, $Q^{\mathcal{A}}$ a finite set of *states*, $q_{\mathsf{I}}^{\mathcal{A}} \in Q^{\mathcal{A}}$ an *initial state*, $\Delta^{\mathcal{A}} \subseteq Q^{\mathcal{A}} \times \Sigma \times Q^{\mathcal{A}} \times Q^{\mathcal{A}}$ a *transition relation*; and $\Omega^{\mathcal{A}}: Q^{\mathcal{A}} \rightarrow C$ a *priority mapping*. An element $(q, a, q_{\mathsf{L}}, q_{\mathsf{R}}) \in \Delta^{\mathcal{A}}$ is called a *transition* of the automaton \mathcal{A} . We say that such a transition is *from* the state q and is *over* the letter a . We make a proviso that, unless stated otherwise, all automata in consideration are *trimmed*, that is, for each state $q \in Q^{\mathcal{A}}$ and letter $a \in \Sigma$, there is at least one transition from q over a in $\Delta^{\mathcal{A}}$. When an automaton \mathcal{A} is known from the context then we skip the superscript and write just Q , Δ , etc.

We extend the notions of parity accepting (resp. rejecting) sequences of priorities to sequences of states by applying Ω , i.e. a sequence of states $(q_n)_{n \in \omega}$ is *parity accepting* (resp. *parity rejecting*) in \mathcal{A} if the priorities $(\Omega(q_n))_{n \in \omega}$ are parity accepting (resp. parity rejecting).

Given a tree $t \in \text{Tr}_\Sigma$, a *run* of an automaton \mathcal{A} over t is a tree $\rho \in \text{Tr}_Q$ such that $\rho(\epsilon) = q_{\mathsf{I}}$ and, for each node $u \in \{\mathsf{L}, \mathsf{R}\}^*$, the tuple

$$(\rho(u), t(u), \rho(u_{\mathsf{L}}), \rho(u_{\mathsf{R}}))$$

is a transition of \mathcal{A} . Such a run is *accepting* if, for every branch $b \in \{\mathsf{L}, \mathsf{R}\}^\omega$, the sequence of states $q_n \stackrel{\text{def}}{=} \rho(b|_n)$ for $n = 0, 1, \dots$ is *parity accepting*. The set of trees over which a given automaton \mathcal{A} has some accepting run is called the *language* of \mathcal{A} and is denoted $L(\mathcal{A})$. If $q \in Q$ is a state of an automaton \mathcal{A} then by $L(\mathcal{A}, q)$ we denote the language of the automaton \mathcal{A} with the initial state set to q . We say that q is *productive* if $L(\mathcal{A}, q) \neq \emptyset$. Thus $L(\mathcal{A}) \neq \emptyset$ iff the initial state $q_{\mathsf{I}}^{\mathcal{A}}$ is productive.

A set of trees is a *regular tree language* if it is of the form $L(\mathcal{A})$ for some non-deterministic automaton \mathcal{A} .

An automaton \mathcal{A} is *deterministic* if for every state $q \in Q$ and letter $a \in \Sigma$ there exists a unique transition in $\Delta^{\mathcal{A}}$ starting from q over a . A deterministic automaton has exactly one run over each tree $t \in \text{Tr}_\Sigma$ and this run can be constructed inductively in a top-down way.

Guidable automata

The notion of a guiding function and one automaton guiding another was introduced in [7], here we follow the exposition from [14].

Fix two non-deterministic automata \mathcal{A} and \mathcal{B} over the same alphabet Σ . A *guiding function* from \mathcal{B} to \mathcal{A} is a function $g: Q^{\mathcal{A}} \times \Delta^{\mathcal{B}} \rightarrow \Delta^{\mathcal{A}}$ such that $g(p, (q, a, q_L, q_R)) = (p, a, p_L, p_R)$ for some $p_L, p_R \in Q^{\mathcal{A}}$ (i.e. the function g is compatible with the state p and the letter a). If $\rho \in \text{Tr}_{Q^{\mathcal{B}}}$ is a run of \mathcal{B} over a tree $t \in \text{Tr}_{\Sigma}$ then we define the tree $\vec{g}(\rho) \in \text{Tr}_{Q^{\mathcal{A}}}$, inductively as follows. We let $\vec{g}(\rho)(\epsilon) \stackrel{\text{def}}{=} q_1^{\mathcal{A}}$ and, for each $u \in \{\text{L}, \text{R}\}^*$, if $\vec{g}(\rho)(u) = p \in Q^{\mathcal{A}}$, $t(u) = a$, and γ is a transition of \mathcal{B} taken in u , i.e. $\gamma \stackrel{\text{def}}{=} (\rho(u), a, \rho(u_L), \rho(u_R))$, and if finally $g(p, \gamma) = (p, a, p_L, p_R) \in \Delta^{\mathcal{A}}$, then we let,

$$\vec{g}(\rho)(u_L) \stackrel{\text{def}}{=} p_L, \quad \vec{g}(\rho)(u_R) \stackrel{\text{def}}{=} p_R.$$

Notice that directly by the definition, the tree $\vec{g}(\rho)$ is a run of \mathcal{A} over t . We say that a guiding function $g: Q^{\mathcal{A}} \times \Delta^{\mathcal{B}} \rightarrow \Delta^{\mathcal{A}}$ *preserves acceptance* if whenever ρ is an accepting run of \mathcal{B} then $\vec{g}(\rho)$ is an accepting run of \mathcal{A} . We say that an automaton \mathcal{B} *guides* an automaton \mathcal{A} (denoted $\mathcal{B} \hookrightarrow \mathcal{A}$), if there exists a guiding function $g: Q^{\mathcal{A}} \times \Delta^{\mathcal{B}} \rightarrow \Delta^{\mathcal{A}}$ which preserves acceptance. In particular, it implies that $L(\mathcal{B}) \subseteq L(\mathcal{A})$.

An automaton \mathcal{A} is *guidable* if it can be guided by any automaton \mathcal{B} such that $L(\mathcal{B}) = L(\mathcal{A})$ (in fact one can equivalently require that $L(\mathcal{B}) \subseteq L(\mathcal{A})$, see [14, Remark 4.5]).

The main result concerning guidable automata is the following theorem.

► **Theorem 1** ([7, Theorem 1], see also [14, Theorem 4.7]). *For every regular tree language L there exists a guidable automaton recognising L . Moreover, such an automaton can be effectively constructed from any non-deterministic automaton for L .*

Since we will rely on the exact structure of such an automaton, we recall the crucial steps in the construction. First, let \mathcal{A} be a non-deterministic automaton recognising the given language L . Assume that $\mathcal{A}' = \langle \Sigma, Q', q'_1, \Delta', \Omega' \rangle$ is any automaton recognising the complement of L .

An important role in this construction is played by *selectors*, i.e. functions $f: \Delta' \rightarrow \{\text{L}, \text{R}\}$ (in other words $f \in \{\text{L}, \text{R}\}^{\Delta'}$). Consider a product tree $t \otimes \tau \in \text{Tr}_{\Sigma \times \{\text{L}, \text{R}\}^{\Delta'}}$ and one of its paths $((a_n, f_n), d_n)_{n \in \omega}$. We say that this path is *losing* if there exists a sequence of transitions $(\delta'_n = (p_n, a_n, p_{L,n}, p_{R,n}))_{n \in \omega}$ of \mathcal{A}' which is *\mathcal{A}' -accepting* in the following sense:

- $p_0 = q_1^{\mathcal{A}'}$,
- $\forall n \in \omega. p_{n+1} = p_{d_n, n} \wedge d_n = f_n(\delta'_n)$,
- the sequence of states $(p_n)_{n \in \omega}$ is parity accepting in \mathcal{A}' .

Lemma 1.15 in [14] provides a deterministic automaton $\text{Winning}(\mathcal{A}')$ over the alphabet $\Sigma \times \{\text{L}, \text{R}\}^{\Delta'}$. The crucial properties of this automaton are stated in the following fact.

► **Fact 2.** *The automaton $\text{Winning}(\mathcal{A}')$ accepts a product tree $t \otimes \tau \in \text{Tr}_{\Sigma \times \{\text{L}, \text{R}\}^{\Delta'}}$ if and only if none of its paths is losing. The projection of $L(\text{Winning}(\mathcal{A}'))$ onto Σ is the original language L , i.e. the complement of $L(\mathcal{A}')$.*

By $\text{Complement}(\mathcal{A}')$ we denote the non-deterministic automaton obtained from the automaton $\text{Winning}(\mathcal{A}')$ by projecting the transitions of $\text{Winning}(\mathcal{A}')$ onto the coordinate Σ : each transition of the form $(q, (a, f), q_L, q_R)$ is replaced by (q, a, q_L, q_R) ; yielding a non-deterministic automaton recognising L . We say that a transition (q, a, q_L, q_R) of $\text{Complement}(\mathcal{A}')$ is *arising* from $f \in \{\text{L}, \text{R}\}^{\Delta'}$ if $(q, (a, f), q_L, q_R)$ is a transition of $\text{Winning}(\mathcal{A}')$.

Notice that in fact the only non-determinism of the automaton $\text{Complement}(\mathcal{A}')$ comes from the projection from the alphabet $\Sigma \times \{\text{L}, \text{R}\}^{\Delta'}$ onto Σ . Therefore, to define a guiding function to guide the automaton $\text{Complement}(\mathcal{A}')$, whenever specifying a transition of $\text{Complement}(\mathcal{A}')$, it is enough to provide a selector $f \in \{\text{L}, \text{R}\}^{\Delta'}$ and then take the unique transition arising from f .

► **Lemma 3** (See the proof of [14, Theorem 4.7]). *The automaton $\text{Complement}(\mathcal{A}')$ is guidable.*

3 Guidability relation

The guidability relation $\mathcal{B} \hookrightarrow \mathcal{A}$ can be seen as a reduction, showing that one automaton uses *less non-determinism* than the other one. Thus, one would naturally expect this relation to be transitive, which is indeed the case.

► **Proposition 4.** *If $\mathcal{C} \hookrightarrow \mathcal{B}$ and $\mathcal{B} \hookrightarrow \mathcal{A}$ then $\mathcal{C} \hookrightarrow \mathcal{A}$.*

A proof of this fact is implicit in [14, Proposition 4.11], where it is shown how to compose two guiding functions $g^{\mathcal{B}}$ and $g^{\mathcal{A}}$ into a winning strategy in a game characterising guidability, called *weak inclusion game*, see [14, page 78]. However, the transitivity of the relation \hookrightarrow has not been explicitly stated before. Therefore, for the sake of completeness, we provide a proof of Proposition 4 here.

First, we recall the notion of the *weak inclusion game* $\mathcal{G}_{\text{guide}}(\mathcal{B}, \mathcal{A})$ from [14, page 78] which is used to characterise when an automaton \mathcal{B} guides another automaton \mathcal{A} , and consequently to decide whether a non-deterministic automaton is guidable. We introduce a small modification, justified below, using the following concept: a transition $(q, a, q_{\text{L}}, q_{\text{R}})$ is *productive* if both states $q_{\text{L}}, q_{\text{R}}$ are productive (cf. Section 2).

The positions of the game $\mathcal{G}_{\text{guide}}(\mathcal{B}, \mathcal{A})$ are of the form $(p, q) \in Q^{\mathcal{A}} \times Q^{\mathcal{B}}$. The initial position (p_0, q_0) is $(q_1^{\mathcal{A}}, q_1^{\mathcal{B}})$. At an n th round for $n = 0, 1, \dots$ which starts in a position (p_n, q_n) :

1. \forall chooses a letter $a_n \in \Sigma$,
2. \forall chooses a productive transition $\gamma_n = (q_n, a_n, q_{\text{L},n}, q_{\text{R},n}) \in \Delta^{\mathcal{B}}$; if there is no productive transition from q_n over a_n then \forall loses immediately,
3. \exists chooses a transition $\delta_n = (p_n, a_n, p_{\text{L},n}, p_{\text{R},n}) \in \Delta^{\mathcal{A}}$,
4. \forall chooses a direction $d_n \in \{\text{L}, \text{R}\}$.

The next position (p_{n+1}, q_{n+1}) of the game is $(p_{d_n,n}, q_{d_n,n})$.

The winning condition for \exists expresses, that if the sequence of states $(q_n)_{n \in \omega}$ is parity accepting in \mathcal{B} then the sequence of states $(p_n)_{n \in \omega}$ is parity accepting in \mathcal{A} .

► **Lemma 5** ([14, Proposition 4.9]). *The player \exists wins the above game $\mathcal{G}_{\text{guide}}(\mathcal{B}, \mathcal{A})$ if and only if $\mathcal{B} \hookrightarrow \mathcal{A}$. Moreover, the player \exists has a positional winning strategy.*

The requirement that the transitions chosen by \forall in step 2 are productive is added because of the case when $L(\mathcal{B}) = \emptyset$. In this corner case the relation $\mathcal{B} \hookrightarrow \mathcal{A}$ holds trivially whereas the characterisation as stated in [14, Proposition 4.9] seems to fail. For the sake of completeness we recall the proof of Proposition 4.9 from [14], taking into account our modification.

Proof. The fact that the game is determined and the player \exists has a positional winning strategy follows from the general theory of infinite games (see, e.g. [21, 12]); indeed the winning criterion of $\mathcal{G}_{\text{guide}}(\mathcal{B}, \mathcal{A})$ can be easily presented as a union of parity criteria and hence it is a so-called Rabin criterion. Clearly, the fact that \exists can also win in finite time does not affect the positional determinacy for this player.

Now if \mathcal{B} guides \mathcal{A} then an acceptance-preserving guiding function $g: Q^{\mathcal{A}} \times \Delta^{\mathcal{B}} \rightarrow \Delta^{\mathcal{A}}$ yields a positional strategy for \exists : in a position (p_n, q_n) , given the choice by \forall of a transition $\gamma_n = (q_n, a_n, q_{L,n}, q_{R,n}) \in \Delta^{\mathcal{B}}$, \exists chooses $\delta_n = g(p_n, \gamma_n)$. Conversely, a positional winning strategy of \exists induces a partial function $\tilde{g}: Q^{\mathcal{A}} \times \Delta^{\mathcal{B}} \rightarrow \Delta^{\mathcal{A}}$ that is defined for $p \in Q^{\mathcal{A}}$ and $\gamma = (q, a, q_L, q_R) \in \Delta^{\mathcal{B}}$ whenever the position (p, q) is winning for \exists and the transition γ is productive. We extend \tilde{g} to a total guiding function $g: Q^{\mathcal{A}} \times \Delta^{\mathcal{B}} \rightarrow \Delta^{\mathcal{A}}$ that for all other arguments is defined in any way; note that it is always possible since, by our proviso, the automaton \mathcal{A} is trimmed. By the assumption, the initial position $(q_I^{\mathcal{A}}, q_I^{\mathcal{B}})$ is winning. Then the fact that \tilde{g} has been derived from a winning strategy implies that the function g preserves acceptance. \blacktriangleleft

► **Remark 6.** Based on the above lemma, the notion that a guiding function *preserves acceptance* can be equivalently rephrased as follows. Consider any sequence of productive transitions $(\gamma_n = (q_n, a_n, q_{L,n}, q_{R,n}))_{n \in \omega}$ of \mathcal{B} and a sequence of directions $(d_n)_{n \in \omega}$. Assume that $q_0 = q_I^{\mathcal{B}}$ and $q_{n+1} = q_{d_n, n}$ for $n = 0, 1, \dots$. Let $\delta_n = (p_n, a_n, p_{L,n}, p_{R,n})$ be defined inductively for $n = 0, 1, \dots$ by $p_0 \stackrel{\text{def}}{=} q_I^{\mathcal{A}}$, $\delta_n = g(p_n, \gamma_n)$, and $p_{n+1} = p_{d_n, n}$. Then the condition that g *preserves acceptance* boils down to saying that if $(q_n)_{n \in \omega}$ is parity accepting in \mathcal{B} then $(p_n)_{n \in \omega}$ is parity accepting in \mathcal{A} .

We can now move to a proof of Proposition 4. Assume that $g^{\mathcal{B}}$ and $g^{\mathcal{A}}$ are two guiding functions which preserve acceptance, witnessing that $\mathcal{C} \leftrightarrow \mathcal{B}$ and $\mathcal{B} \leftrightarrow \mathcal{A}$.

We will show that \exists has a winning strategy in $\mathcal{G}_{\text{guide}}(\mathcal{C}, \mathcal{A})$, using the set of states of \mathcal{B} as a memory structure. More precisely, \exists keeps track of an additional state $r_n \in Q^{\mathcal{B}}$ with $r_0 \stackrel{\text{def}}{=} q_I^{\mathcal{B}}$. Consider an n th round starting in a position (p_n, q_n) with a memory state r_n of \exists . Assume that \forall has played $a_n \in \Sigma$ and $\gamma_n \in \Delta^{\mathcal{C}}$ as above. Let $\gamma'_n = (r_n, a_n, r_{L,n}, r_{R,n}) \stackrel{\text{def}}{=} g^{\mathcal{B}}(r_n, \gamma_n) \in \Delta^{\mathcal{B}}$ be the transition of \mathcal{B} given by $g^{\mathcal{B}}$. Note that if the transition γ_n was productive then γ'_n must be productive as well. Similarly, let $\delta_n \stackrel{\text{def}}{=} g^{\mathcal{A}}(p_n, \gamma'_n)$ be the transition of \mathcal{A} given by $g^{\mathcal{A}}$. Let \exists play δ_n as her choice in that round. Once the round is finished, let the new memory state of \exists be $r_{n+1} \stackrel{\text{def}}{=} r_{d_n, n}$.

► **Claim 7.** The strategy defined above is in fact winning for \exists .

Proof. Consider a play of $\mathcal{G}_{\text{guide}}(\mathcal{C}, \mathcal{A})$ that was played according to the above strategy. Let $(r_n)_{n \in \omega}$ be the sequence of memory states of \exists used during that play. Assume that $(q_n)_{n \in \omega}$ is parity accepting in \mathcal{C} . Since $g^{\mathcal{B}}$ represents a winning strategy of \exists in $\mathcal{G}_{\text{guide}}(\mathcal{C}, \mathcal{B})$, we know that $(r_n)_{n \in \omega}$ is parity accepting in \mathcal{B} . Therefore, we can use the fact that $g^{\mathcal{A}}$ represents a winning strategy of \exists in $\mathcal{G}_{\text{guide}}(\mathcal{B}, \mathcal{A})$ and entail that $(p_n)_{n \in \omega}$ is parity accepting in \mathcal{A} . \triangleleft

This concludes the proof of Proposition 4.

► **Corollary 8** (See [14, Proposition 4.11]). *Consider an automaton \mathcal{A} and a guidable automaton \mathcal{B} , both recognising the same language L . Then the automaton \mathcal{A} is guidable if and only if $\mathcal{B} \leftrightarrow \mathcal{A}$.*

Note that the above corollary combined with Lemma 5 and Theorem 1 yield a procedure to decide whether a non-deterministic automaton is guidable ([14, Theorem 4.7]).

4 Game automata as guidable automata

It is straightforward to see that deterministic automata are guidable (the function g in the definition above does not depend on the argument in $\Delta^{\mathcal{B}}$). Hence, guidable automata can be viewed as a semantic extension of deterministic automata. However, there already

exists an established notion of automata naturally extending the deterministic ones, namely the *game automata*, see [8]. In this section we recall this notion in the framework of non-deterministic automata, and prove the following new result.

► **Proposition 9.** *Every game automaton is guidable.*

Since every deterministic automaton is also game, we get a syntactic stratification:

$$\text{Deterministic} \subsetneq \text{Game} \subsetneq \text{Guidable}, \quad (1)$$

with each consecutive class not only containing more automata but also recognising more languages. Theorem 1 means that the last class in this stratification is expressively complete for all regular tree languages.

We say that a non-deterministic tree automaton \mathcal{A} is a *game automaton* if it satisfies the following two properties. First of all, Q contains a non-initial all-accepting state $\top \in Q$: $\Omega(\top)$ is even and for each letter $a \in A$ there is a unique transition starting from \top over a , namely (\top, a, \top, \top) (thus $L(\mathcal{A}, \top) = \text{Tr}_\Sigma$). Moreover, for a state $q \in Q - \{\top\}$ and a letter $a \in \Sigma$ we say that (q, a) is in one of the following *modes*:

- **conjunctive:** there is a unique transition starting from q over a of the form (q, a, q_L, q_R) with both q_L and q_R distinct from \top ;
- **disjunctive:** there are exactly two transitions starting from q over a , one of the form (q, a, q_L, \top) and the other of the form (q, a, \top, q_R) , with both q_L and q_R distinct from \top .

The above definition is a direct translation of the alternating formulation of game automata into the format of non-deterministic automata. The two modes above correspond to the use of \wedge or \vee in the transition formula $\delta(q, a)$, see [11, Definition 3.2].

Before we move to a proof of Proposition 9, we need to recall that game automata admit a syntactic complementation construction. Fix a game automaton $\mathcal{A} = \langle \Sigma, Q, q_I, \Delta, \Omega \rangle$. Consider another game automaton denoted \mathcal{A}^c defined as $\mathcal{A}^c = \langle \Sigma, Q, q_I, \Delta', \Omega' \rangle$, where Δ' and Ω' are defined as follows. First of all, $\Omega'(q) \stackrel{\text{def}}{=} \Omega(q) + 1$ for all q distinct than \top and $\Omega'(\top) \stackrel{\text{def}}{=} \Omega(\top)$. Moreover, Δ' contains the following transitions:

- for each $a \in \Sigma$ we have $(\top, a, \top, \top) \in \Delta'$;
- if $q \in Q - \{\top\}$ and $a \in \Sigma$ are in conjunctive mode and $(q, a, q_L, q_R) \in \Delta$ then the mode of (q, a) in \mathcal{A}^c is disjunctive, with (q, a, q_L, \top) and (q, a, \top, q_R) in Δ' ;
- if $q \in Q - \{\top\}$ and $a \in \Sigma$ are in disjunctive mode and (q, a, q_L, \top) , $(q, a, \top, q_R) \in \Delta$ then the mode of (q, a) in \mathcal{A}^c is conjunctive, with $(q, a, q_L, q_R) \in \Delta'$.

The following fact is an immediate consequence of the alternating semantics of game automata, see [11, page 24:7].

► **Fact 10.** *The automaton \mathcal{A}^c is also a game automaton and $L(\mathcal{A}^c) = \text{Tr}_\Sigma - L(\mathcal{A})$.*

The rest of this section is devoted to a proof of Proposition 9. Fix a game automaton $\mathcal{A} = \langle \Sigma, Q, q_I, \Delta, \Omega \rangle$ and let $\mathcal{A}' \stackrel{\text{def}}{=} \mathcal{A}^c$ denote its syntactic complement defined as above. Let Δ' be the set of transitions of \mathcal{A}' . Notice that (up to subtracting 2 from Ω) the automaton $(\mathcal{A}')^c$ is equal to the original automaton \mathcal{A} . We will now apply the procedure of constructing a guidable automaton for L . Recall that this procedure involves a deterministic automaton $\text{Winning}(\mathcal{A}')$ that accepts a product tree $t \otimes \tau \in \text{Tr}_{\Sigma \times \{\text{L,R}\}^{\Delta'}}$ if and only if none of its paths is losing. The desired guidable automaton $\text{Complement}(\mathcal{A}')$ is then obtained from $\text{Winning}(\mathcal{A}')$ by projection on the component Σ . The automaton $\text{Winning}(\mathcal{A}')$ itself is not concretely specified; it can be any deterministic automaton with the required property. We

will define it in such a way that the eventual automaton $\text{Complement}(\mathcal{A}')$ will coincide with the original automaton \mathcal{A} . It is convenient here to use a variant of a deterministic automaton where transitions leading to non-productive states are undefined, so that there is always *at most one* transition rather than exactly one. (So we momentarily suspend our proviso that all automata are trimmed, but the projection automaton will be trimmed again.) Let $\mathcal{A}_w = \langle \Sigma \times \{\mathsf{L}, \mathsf{R}\}^{\Delta'}, Q, q_I, \Delta_w, \Omega \rangle$; that is, the automaton shares the states and the priority function with \mathcal{A} . The transitions of \mathcal{A}_w are defined as follows.

- For each $a \in \Sigma$ and $f \in \{\mathsf{L}, \mathsf{R}\}^{\Delta'}$, we have $(\top, (a, f), \top, \top) \in \Delta_w$.
- If (q, a) are in disjunctive mode in \mathcal{A} , and $(q, a, q_L, \top), (q, a, \top, q_R) \in \Delta$, which implies that $(q, a, q_L, q_R) \in \Delta'$, and if $f: (q, a, q_L, q_R) \mapsto d$ (for $f \in \{\mathsf{L}, \mathsf{R}\}^{\Delta'}$) then the unique transition of \mathcal{A}_w from q over (a, f) is $(q, (a, f), q_L, \top)$ or $(q, (a, f), \top, q_R)$ depending on whether $d = \mathsf{L}$ or $d = \mathsf{R}$.
- If (q, a) are in conjunctive mode in \mathcal{A} , and $(q, a, q_L, q_R) \in \Delta$, which implies that $(q, a, q_L, \top), (q, a, \top, q_R) \in \Delta'$, and if moreover the selector f “behaves properly” in the sense that $f: (q, a, q_L, \top) \mapsto \mathsf{L}$ and $f: (q, a, \top, q_R) \mapsto \mathsf{R}$ then the unique transition of \mathcal{A}_w from q over (a, f) is $(q, (a, f), q_L, q_R)$. In all other cases, a transition from q over (a, f) is undefined.

Now, it is clear that an automaton $\text{Complement}(\mathcal{A}_w)$ obtained from \mathcal{A} by replacing each transition $(q, (a, f), q_L, q_R)$ by (q, a, q_L, q_R) (cf. Section 2) coincides with the original automaton \mathcal{A} . So the proof of Proposition 9 boils down to the following.

▷ **Claim 11.** The automaton \mathcal{A}_w accepts a product tree $t \otimes \tau \in \text{Tr}_{\Sigma \times \{\mathsf{L}, \mathsf{R}\}^{\Delta'}}$ if and only if none of its paths is losing.

Proof. Let the automaton \mathcal{A}_w accept a product tree $t \otimes \tau \in \text{Tr}_{\Sigma \times \{\mathsf{L}, \mathsf{R}\}^{\Delta'}}$ by a unique run τ . For the sake of contradiction, suppose that there exists a losing path $((a_n, f_n), d_n)_{n \in \omega}$ along with a sequence of transitions $(\delta'_n = (p_n, a_n, p_{\mathsf{L}, n}, p_{\mathsf{R}, n}))_{n \in \omega}$ of \mathcal{A}' , such that $p_0 = q_I^{\mathcal{A}'}$, and $\forall n \in \omega, p_{n+1} = p_{d_n, n} \wedge d_n = f_n(\delta'_n)$. Let $(q_n)_{n \in \omega}$ be the sequence of states visited by the run τ on this path, i.e. $q_n = \tau(d_0 \dots d_{n-1})$. We verify by induction on n that $q_n = p_n \neq \top$. Clearly $q_n = q_I^{\mathcal{A}} = q_I^{\mathcal{A}'} = p_0 \neq \top$. Suppose the claim holds for n . If (q_n, a_n) is in conjunctive mode in \mathcal{A} with the unique transition (q_n, a_n, q_L, q_R) (with $q_L, q_R \neq \top$) then (p_n, a_n) (with $p_n = q_n$) is in disjunctive mode in \mathcal{A}' , with two possible transitions. Let $\delta'_n = (p_n, a_n, p_{\mathsf{L}, n}, p_{\mathsf{R}, n}) = (q_n, a_n, q_L, \top)$; the other case is symmetric. It follows from the definition of \mathcal{A}_w that in this case $f_n: \delta'_n \mapsto \mathsf{L}$, as otherwise the next transition would not be defined and the run τ would not be accepting. Clearly, the transition of \mathcal{A}_w used at this point is $(q_n, (a_n, f_n), q_L, q_R)$. Therefore, we have $d_{n+1} = \mathsf{L}$ and $q_{n+1} = q_L = p_{n+1} \neq \top$, as required. If (q_n, a_n) is in disjunctive mode in \mathcal{A} and hence $\delta'_n = (p_n, a_n, p_{\mathsf{L}, n}, p_{\mathsf{R}, n})$ is the unique transition of \mathcal{A}' from $p_n = q_n$ over a_n , and moreover $f_n: \delta'_n \mapsto d_n$ then the transition of \mathcal{A}_w from q_n over (a_n, f_n) is $(p_n, (a_n, f_n), p_{\mathsf{L}, n}, \top)$ or $(q, (a_n, f_n), \top, p_{\mathsf{R}, n})$ depending on whether $d_n = \mathsf{L}$ or $d_n = \mathsf{R}$, but in any case we have $q_{n+1} = q_L = p_{n+1} \neq \top$, as required.

As the sequence of states $(q_n)_{n \in \omega}$ is parity accepting in \mathcal{A}_w (hence also in \mathcal{A}), it cannot be at the same time parity accepting in \mathcal{A}' , which yields a contradiction.

Now let $t \otimes \tau \in \text{Tr}_{\Sigma \times \{\mathsf{L}, \mathsf{R}\}^{\Delta'}}$ be a product tree not accepted by \mathcal{A}_w . There are two possibilities: either the unique run of \mathcal{A}_w on $t \otimes \tau$ is non-accepting, or there is no (complete) run at all, because the transitions are blocked at some place. We will show that in both cases there is a losing path in $t \otimes \tau$. Suppose first that in a run ρ of \mathcal{A}_w on $t \otimes \tau$ there is a path $(q_n, d_n)_{n \in \omega}$, with $q_0 = q_I^{\mathcal{A}}$ and $q_{n+1} = \rho(d_0 \dots d_n)$, such that the sequence $(q_n)_{n \in \omega}$ is parity rejecting. Thus in particular $q_n \neq \top$, for all n . Let $a_n = t(d_0 \dots d_{n-1})$ and $f_n = \tau(d_0 \dots d_{n-1})$. We will define inductively a sequence of transitions $(\delta'_n = (p_n, a_n, p_{\mathsf{L}, n}, p_{\mathsf{R}, n}))_{n \in \omega}$ of \mathcal{A}'

with $p_n = q_n$, which along with the sequence $(d_n)_{n \in \omega}$ will witness a losing path. As usual, we consider two cases. If (q_n, a_n) are in disjunctive mode in \mathcal{A} then we let $\delta'_n \stackrel{\text{def}}{=} (q_n, a_n, q_L, q_R)$, where the last is the unique transition of \mathcal{A}' from q_n over a_n . Suppose further that $f_n: (q_n, a_n, q_L, q_R) \mapsto \mathbb{L}$ (the other case is symmetric). Then the transition used by the automaton \mathcal{A}_w is $(q_n, (a_n, f_n), q_L, \top)$, and we know that $q_{n+1} = q_L$ (because it is different from \top), hence $d_n = \mathbb{L}$ and the local condition of a losing path is satisfied. If (q_n, a_n) are in conjunctive mode in \mathcal{A} with the transition (q_n, a_n, q_L, q_R) then, since the transition of \mathcal{A}_w over (a_n, f_n) is defined, we know that f_n “behaves properly”, i.e. $f_n: (q_n, a_n, q_L, \top) \mapsto \mathbb{L}$ and $f: (q_n, a_n, \top, q_R) \mapsto \mathbb{R}$. Suppose $d_n = \mathbb{L}$ (the other case is symmetric), hence $q_{n+1} = q_L$. We let $\delta'_n \stackrel{\text{def}}{=} (q_n, a_n, q_L, \top)$; then again the local condition of a losing path is satisfied. Thus we have obtained a losing path in $t \otimes \tau$, as expected.

Now suppose that there is no run of \mathcal{A}_w on $t \otimes \tau$. We may however define a *partial run*, i.e. a mapping $\rho: \text{dom} \rightarrow Q$, where $\text{dom} \subseteq \{\mathbb{L}, \mathbb{R}\}^*$ is closed under initial segment, such that $\rho(\epsilon) = q_I^A$, and whenever $\tau(v) = q$ and the transition $(q, (t(v), \rho(v)), q_L, q_R)$ of \mathcal{A}_w is defined then $\rho(v\mathbb{L}) = q_L$ and $\rho(v\mathbb{R}) = q_R$. Since ρ is not complete, there must be a finite path d_0, \dots, d_{m-1} , such that $\rho(d_0, \dots, d_{m-1}) = q$, $t(d_0, \dots, d_{m-1}) = a$, $\tau(d_0, \dots, d_{m-1}) = f$, and there is no transition of \mathcal{A}_w from q over (a, f) . (It is possible that $m = 0$.) This means that (q, a) are in conjunctive mode in \mathcal{A} with the transition (q, a, q_L, w_R) , but f does not “behave properly”; for example $f: (q, a, q_L, \top) \mapsto \mathbb{R}$ (the other case is symmetric). Let $q_i = \rho(d_0 \dots d_{i-1})$, for $i = 0, \dots, m$. We define a finite sequence $\delta'_0, \dots, \delta'_{m-1}$ of transitions of \mathcal{A}' corresponding to the path d_0, \dots, d_{m-1} , exactly as in the previous case, satisfying the local condition of a losing path, with $p_i = q_i$, and hence $p_m = q_m = q$. Now, under the assumption that $f: (q, a, q_L, \top) \mapsto \mathbb{R}$, we let $\delta'_m \stackrel{\text{def}}{=} (q, a, q_L, \top)$, $d_m = \mathbb{R}$, and $p_{m+1} = \top$. Then we can clearly prolong this path to a losing path (with δ'_n , for $n \geq m+1$ being a transition for \top).

This concludes the proof of Claim 11, and hence also of Proposition 9. \triangleleft

5 Decidability of guidable index

In this section we provide the main result of the present article:

► **Theorem 12.** *Given a regular language of infinite trees $L \subseteq \text{Tr}_\Sigma$ and an index $C = \{i, \dots, j\}$ it is decidable whether there exists a guidable C -parity automaton which recognises L .*

Let $\mathcal{A} = \langle \Sigma, Q^A, q_I^A, \Delta^A, \Omega^A \rangle$ be a guidable parity automaton for the given language L , which can be constructed as in Theorem 1.

Consider the following game, played between two players called \exists and \forall . The positions of the game are pairs of states $Q^A \times Q^A$ of the automaton \mathcal{A} , the initial position (p_0, p'_0) is (q_I^A, q_I^A) . At an n th round for $n = 0, 1, \dots$ which starts in a position $(p_n, p'_n) \in Q^A \times Q^A$:

1. \exists chooses a priority $c_n \in C$,
2. \forall chooses a transition $\delta_n = (p_n, a_n, p_{L,n}, p_{R,n}) \in \Delta^A$ from p_n over some letter $a_n \in \Sigma$,
3. \exists chooses a transition $\delta'_n = (p'_n, a_n, p'_{L,n}, p'_{R,n}) \in \Delta^A$ from p'_n over the same letter $a_n \in \Sigma$,
4. \forall chooses a direction $d_n \in \{\mathbb{L}, \mathbb{R}\}$.

The next position of the game (p_{n+1}, p'_{n+1}) is $(p_{d_n, n}, p'_{d_n, n})$.

The winning condition for \exists in that game is the conjunction of the following two conditions:

- W1** If the sequence of states $(p_n)_{n \in \omega}$ is parity accepting in \mathcal{A} then the sequence of priorities $(c_n)_{n \in \omega}$ must be also parity accepting.
- W2** If the sequence of priorities $(c_n)_{n \in \omega}$ is parity accepting then the sequence of states $(p'_n)_{n \in \omega}$ is parity accepting in \mathcal{A} .

Clearly the winning condition of the game is ω -regular so the winner effectively has a finite memory winning strategy [3]. The rest of this section is devoted to a proof of the following proposition.

► **Proposition 13.** *The player \exists wins the game above if and only if there exists a guidable C -parity automaton \mathcal{B} which recognises the language $L = L(\mathcal{A})$.*

► **Corollary 14.** *The problem of existence of a guidable C -parity automaton recognising a given regular language is decidable. Moreover, it is possible to effectively construct such an automaton whenever it exists.*

The construction of a guidable automaton for the language recognised by a given non-deterministic automaton may involve a double-exponential growth in the number of states [14, Proposition 4.8]. Therefore, already the size of the above game is in general double-exponential in the given representation of L . Since the index of the constructed automaton \mathcal{A} is only single-exponential, there exists a double-exponential deterministic parity automaton with single-exponentially many priorities verifying the conjunction of the winning conditions W1 and W2 (cf. [6, Section 7]). It all implies that the complexity of the proposed algorithm is 2-EXPTIME.

Soundness

Assume that \exists has a (finite memory) winning strategy in the game above. Such a strategy consists of a finite set of memory values M ; the initial memory value $m_I \in M$; a memory update function of the type:

$$\tau: Q^{\mathcal{A}} \times Q^{\mathcal{A}} \times M \times \Delta^{\mathcal{A}} \times \{\mathsf{L}, \mathsf{R}\} \rightarrow M,$$

which updates the memory value depending on the choices of the opponent in a given round; and two skolemised decision functions:

- $\bar{c}: Q^{\mathcal{A}} \times Q^{\mathcal{A}} \times M \rightarrow C$ which gives the choices of the priorities c_n depending only on the position of the game and the memory value in M ,
- $\bar{\delta}': Q^{\mathcal{A}} \times Q^{\mathcal{A}} \times M \times \Delta^{\mathcal{A}} \rightarrow \Delta^{\mathcal{A}}$ which gives the choices of the transitions δ'_n depending on the position of the game, the memory value in M , and the choice of the transition δ_n made by \forall in the given round.

We construct a non-deterministic automaton \mathcal{B} which guesses the choices of the transitions $\delta_n \in \Delta^{\mathcal{A}}$ and simulates the above strategy. Together with a definition of \mathcal{B} , we define a guiding function $g^{\mathcal{B}}: Q^{\mathcal{B}} \times \Delta^{\mathcal{A}} \rightarrow \Delta^{\mathcal{B}}$ that will be used to witness that $\mathcal{A} \leftrightarrow \mathcal{B}$.

Let the set of states of the automaton \mathcal{B} be $Q^{\mathcal{A}} \times Q^{\mathcal{A}} \times M$. The initial state is $(q_I^{\mathcal{A}}, q_I^{\mathcal{A}}, m_I)$. The priority of a state (p, p', m) is $\bar{c}(p, p', m)$. Given a state (p, p', m) of \mathcal{B} and a transition $\delta = (p, a, p_L, p_R) \in \Delta^{\mathcal{A}}$ we define $g^{\mathcal{B}}((p, p', m), \delta)$ as the transition $\left((p, p', m), a, (p_L, p'_L, m_L), (p_R, p'_R, m_R) \right)$ such that $\bar{\delta}'(p, p', m, \delta) = \delta' = (p', a, p'_L, p'_R) \in \Delta^{\mathcal{A}}$ and for each direction $d \in \{\mathsf{L}, \mathsf{R}\}$ we have $\tau(p, p', m, \delta, d) = m_d$. The set of transitions of \mathcal{B} consists of all the transitions $g^{\mathcal{B}}((p, p', m), \delta)$ for $(p, p', m) \in Q^{\mathcal{B}}$ and $\delta \in \Delta^{\mathcal{A}}$. This concludes the definition of the automaton \mathcal{B} .

In the following two proofs we will rely on the path-wise definition of when a guiding function preserves acceptance, see Remark 6.

The following fact follows directly from the assumption that we began with a winning strategy of \exists and therefore its choices satisfy the winning condition W1.

► **Fact 15.** *The guiding function $g^{\mathcal{B}}$ defined above preserves acceptance. Thus, it witnesses that $\mathcal{A} \hookrightarrow \mathcal{B}$ and in particular $L(\mathcal{A}) \subseteq L(\mathcal{B})$.*

► **Lemma 16.** *We have $\mathcal{B} \hookrightarrow \mathcal{A}$ and in particular $L(\mathcal{B}) \subseteq L(\mathcal{A})$.*

Proof. We can use the decision function $\bar{\delta}^r$ as a guide function $g^{\mathcal{A}}: Q^{\mathcal{A}} \times \Delta^{\mathcal{B}} \rightarrow \Delta^{\mathcal{A}}$. More formally, consider a transition γ of \mathcal{B} that is of the form $g^{\mathcal{B}}((p, p', m), \delta)$ for some¹ $\delta \in \Delta^{\mathcal{A}}$. Let $g^{\mathcal{A}}(p', \gamma) = \bar{\delta}^r(p, p', m, \delta)$; the remaining values of $g^{\mathcal{A}}(p'', \gamma)$ with the first argument p'' different than p' that comes from γ are irrelevant.

The fact that $g^{\mathcal{A}}$ preserves acceptance follows directly from the assumption that we began with a winning strategy of \exists and therefore its choices satisfy the winning condition W2. ◀

The two above observations allow us to use Corollary 8 to learn the following fact.

► **Fact 17.** *The automaton \mathcal{B} is guidable and recognises the language $L = L(\mathcal{A})$.*

Since \mathcal{B} is a C -parity automaton, the above fact concludes this direction of the proof.

Completeness

We will now show that if there exists a guidable C -parity automaton \mathcal{B} which recognises L then \exists has a winning strategy in the game above. Let $\mathcal{B} = \langle \Sigma, Q^{\mathcal{B}}, q_1^{\mathcal{B}}, \Delta^{\mathcal{B}}, \Omega^{\mathcal{B}} \rangle$ be such a guidable automaton.

Fix two guiding functions $g^{\mathcal{A}}: Q^{\mathcal{A}} \times \Delta^{\mathcal{B}} \rightarrow \Delta^{\mathcal{A}}$ and $g^{\mathcal{B}}: Q^{\mathcal{B}} \times \Delta^{\mathcal{A}} \rightarrow \Delta^{\mathcal{B}}$, witnessing that $\mathcal{B} \hookrightarrow \mathcal{A}$ and $\mathcal{A} \hookrightarrow \mathcal{B}$, respectively.

Let the memory structure of the constructed strategy of \exists consists of the set of states of \mathcal{B} . The initial memory value q_0 is $q_1^{\mathcal{B}}$. In an n th round of the game that starts in a position $(p_n, p'_n) \in Q^{\mathcal{A}} \times Q^{\mathcal{A}}$ and with a memory value $q_n \in Q^{\mathcal{B}}$ let \exists play as follows:

- \exists plays the priority $c_n \stackrel{\text{def}}{=} \Omega^{\mathcal{B}}(q_n)$,
- \forall plays a transition $\delta_n = (p_n, a_n, p_{L,n}, p_{R,n}) \in \Delta^{\mathcal{A}}$, which is mapped by $g^{\mathcal{B}}$ to a transition $\gamma = (q_n, a_n, q_{L,n}, q_{R,n}) \stackrel{\text{def}}{=} g^{\mathcal{B}}(q_n, \delta) \in \Delta^{\mathcal{B}}$,
- \exists plays the transition $\delta'_n = (p'_n, a_n, p'_{L,n}, p'_{R,n}) \stackrel{\text{def}}{=} g^{\mathcal{A}}(p'_n, \gamma) \in \Delta^{\mathcal{A}}$,
- \forall plays a direction $d_n \in \{L, R\}$,

and the next memory value q_{n+1} is $q_{d_n, n}$.

The following fact is an immediate consequence of the assumptions that the functions $g^{\mathcal{B}}$ and $g^{\mathcal{A}}$ map accepting runs into accepting runs, see Remark 6.

► **Lemma 18.** *The strategy defined above is winning for \exists .*

Proof. Consider an infinite play of the game where \exists played according to the above defined strategy. First consider the winning condition W1. Assume that the sequence of states $(p_n)_{n \in \omega}$ is parity accepting in \mathcal{A} . Therefore, by the assumption on $g^{\mathcal{B}}$ we know that the sequence of states $(q_n)_{n \in \omega}$ is parity accepting in \mathcal{B} . But the choice of priorities c_n as $\Omega^{\mathcal{B}}(q_n)$ guarantees that the sequence of priorities $(c_n)_{n \in \omega}$ must also be parity accepting.

Now consider the winning condition W2 and assume that the sequence of priorities $(c_n)_{n \in \omega}$ is parity accepting. Therefore, the sequence of states $(q_n)_{n \in \omega}$ must be parity accepting in \mathcal{B} . Based on the assumption on $g^{\mathcal{A}}$ we know that the sequence of states $(p'_n)_{n \in \omega}$ must be parity accepting in \mathcal{A} . ◀

¹ In fact the transition δ is uniquely determined by γ .

6

 Index transfer results

In this section we show results binding deterministic, game, and guidable indices of languages, as expressed by the following proposition.

► **Proposition 19.** *Assume that $L \subseteq \text{Tr}_\Sigma$ can be recognised by some deterministic (resp. game) automaton and by some guidable C -parity automaton. Then L can be recognised by a deterministic (resp. game) C -parity automaton.*

First consider the case of deterministic automata. Let \mathcal{D} be any deterministic automaton recognising L and let \mathcal{A} be a guidable C -parity automaton recognising L . Let $g^{\mathcal{A}}: Q^{\mathcal{A}} \times \Delta^{\mathcal{D}} \rightarrow \Delta^{\mathcal{A}}$ be a guiding function witnessing that $\mathcal{D} \hookrightarrow \mathcal{A}$.

Consider the automaton \mathcal{B} constructed as a product of \mathcal{D} and \mathcal{A} . More formally, the set of states of \mathcal{B} is $Q^{\mathcal{D}} \times Q^{\mathcal{A}}$; the initial state of \mathcal{B} is $(q_{\top}^{\mathcal{D}}, q_{\top}^{\mathcal{A}})$; the priority function is defined as $\Omega^{\mathcal{B}}(q, p) = \Omega^{\mathcal{A}}(p)$; and the transitions of \mathcal{B} are of the form

$$((q, p), a, (q_L, p_L), (q_R, p_R)) \in \Delta^{\mathcal{B}}$$

where $\gamma = (q, a, q_L, q_R)$ is the unique transition of \mathcal{D} from q over a and $\delta = (p, a, p_L, p_R) \stackrel{\text{def}}{=} g^{\mathcal{A}}(p, \gamma) \in \Delta^{\mathcal{A}}$ is the transition given by the guiding function $g^{\mathcal{A}}$. Directly from the definition we see that \mathcal{B} is a deterministic C -parity automaton. Therefore, the following fact concludes the proof of Proposition 19 in the case of deterministic automata.

► **Lemma 20.** $L(\mathcal{B}) = L$.

Proof. The fact that $L(\mathcal{B}) \subseteq L(\mathcal{A})$ is immediate, as each accepting run of \mathcal{B} encodes an accepting run of \mathcal{A} over the same tree. On the other hand, if $t \in L(\mathcal{D})$ then the assumptions on $g^{\mathcal{A}}$ imply that the unique run of \mathcal{B} over t must be accepting. ◀

We now move to the proof of Proposition 19 in the case of game automata. Similarly as before, take any game automaton \mathcal{D} recognising L and let \mathcal{A} be a guidable C -parity automaton recognising L .

We first modify the automaton \mathcal{A} by adding an additional state \top of even priority and a transition of the form (\top, a, \top, \top) for each letter $a \in \Sigma$. Now, for each state $p \in Q^{\mathcal{A}} - \{\top\}$ such that $L(\mathcal{A}, p) = \text{Tr}_\Sigma$, remove this state and replace each occurrence of p by \top in all the transitions of \mathcal{A} . Clearly, these modifications do not change the language recognised by \mathcal{A} . Moreover, if the original automaton was guidable then the new one is also guidable. For the sake of simplicity we assume that from this moment on \mathcal{A} denotes the modified automaton. Let $g^{\mathcal{A}}: Q^{\mathcal{A}} \times \Delta^{\mathcal{D}} \rightarrow \Delta^{\mathcal{A}}$ be a guiding function witnessing that $\mathcal{D} \hookrightarrow \mathcal{A}$.

The automaton \mathcal{B} is defined analogously as in the deterministic case, as a product of the automata \mathcal{D} and \mathcal{A} using the guiding function $g^{\mathcal{A}}$. We additionally restrict the set of states of \mathcal{B} to those that can be reached from the initial state $(q_{\top}^{\mathcal{D}}, q_{\top}^{\mathcal{A}})$ using the transitions of \mathcal{B} .

▷ **Claim 21.** If a state (\top, p) is reachable from the initial state of \mathcal{B} by the transitions of \mathcal{B} then $L(\mathcal{A}, p) = \text{Tr}_\Sigma$, which means that $p = \top$.

Proof. Since all the transitions of \mathcal{B} follow the guiding function $g^{\mathcal{A}}$, whenever a state (q, p) is reachable in the automaton \mathcal{B} , we know that it is a winning position of \exists in the weak inclusion game $\mathcal{G}_{\text{guide}}(\mathcal{D}, \mathcal{A})$, see Lemma 5. This guarantees that $L(\mathcal{D}, q) \subseteq L(\mathcal{A}, p)$, and as $L(\mathcal{D}, \top) = \text{Tr}_\Sigma$, we know that $L(\mathcal{A}, p) = \text{Tr}_\Sigma$. ◀

The above claim implies that \mathcal{B} is in fact a game automaton with the shape of transitions inherited from \mathcal{D} and (\top, \top) playing the role of the state \top of \mathcal{B} . Since the priority function $\Omega^{\mathcal{B}}$ is again inherited from \mathcal{A} , \mathcal{B} is a C -parity automaton.

It remains to show that $L(\mathcal{B}) = L$. Similarly as in the deterministic case, the inclusion $L(\mathcal{B}) \subseteq L(\mathcal{A})$ is immediate as each accepting run of \mathcal{B} encodes an accepting run of \mathcal{A} . On the other hand, the inclusion $L(\mathcal{D}) \subseteq L(\mathcal{B})$ is direct from the assumptions on $g^{\mathcal{A}}$ – each accepting run of \mathcal{D} is mapped by $g^{\mathcal{A}}$ into an accepting run of \mathcal{B} .

This concludes the proof of Proposition 19.

Consequences

By combining Proposition 19 and Proposition 9 (or by direct construction) we can observe the following.

► **Remark 22.** If $L \subseteq \text{Tr}_{\Sigma}$ is recognised by some deterministic automaton and by some game C -parity automaton then L can be recognised by a deterministic C -parity automaton.

Thus the stratification from (1) preserves indices of the languages: if a language happens to be recognisable by a less expressive automaton then its parity index is the same from the point of view of both less and more expressive classes.

On the other hand, it is known that there is no such transfer when moving between deterministic and (general) non-deterministic automata. Indeed, we have the following property on ω -words.

► **Fact 23** ([22]). *For each $C = \{i, \dots, j\} \subseteq \omega$ there exists a regular language of ω -words, which can be recognised by a non-deterministic $\{1, 2\}$ -parity (Büchi) ω -word automaton, but not by any deterministic C -parity ω -word automaton.*

This property can be easily shifted to infinite trees (e.g., by considering the set of all trees whose leftmost branch belongs to a given regular language of ω -words). Thus we obtain the following consequence of Proposition 19.

► **Corollary 24.** *For each $C = \{i, \dots, j\} \subseteq \omega$ there exists a regular tree language L which can be recognised by some deterministic automaton and some non-deterministic $\{1, 2\}$ -parity automaton but not by any deterministic, game, nor guidable C -parity automaton.*

7 Conclusions

The present work is focused on the class of guidable automata. We show that they syntactically extend the previously studied classes of deterministic and game automata. Moreover, we provide an algorithm solving the guidable index problem. Finally, we show that for the three considered classes of deterministic, game, and guidable automata, the index can be transferred. As a negative consequence of this fact (Corollary 24) we show that there is no correspondence between the general non-deterministic index of a regular tree language and its guidable index.

Although these results do not bring a direct progress in the general non-deterministic index problem; we hope that they may be useful in at least two ways.

First, we believe that the class of guidable automata is worth separate attention. As our new results indicate, this class of tree automata is tractable and extends the previously considered classes of *structurally simple* automata: deterministic and game. This is especially important as, contrarily to the other two classes, guidable automata are expressively complete for all regular tree languages.

Second, the present results indicate a new way of approaching the index problems, by providing new game-based techniques. In particular, we believe that the interplay between the two sequences of transitions, δ_n and δ'_n , constructed in the game from Section 5, gives some insight on ways of forcing the players to witness the existence of certain objects.

References

- 1 Julian Bradfield. Simplifying the modal mu-calculus alternation hierarchy. In *STACS*, pages 39–49, 1998.
- 2 Julius Richard Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11, 1962.
- 3 Julius Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.
- 4 Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *49th Annual ACM STOC*, pages 252–263, 2017.
- 5 Lorenzo Clemente and Michał Skrzypczak. Deterministic and game separability for regular languages of infinite trees. accepted to ICALP, 2021.
- 6 Lorenzo Clemente and Michał Skrzypczak. Deterministic and game separability for regular languages of infinite trees, 2021. [arXiv:2105.01137](https://arxiv.org/abs/2105.01137).
- 7 Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP (2)*, pages 398–409, 2008.
- 8 Jacques Duparc, Alessandro Facchini, and Filip Murlak. Linear game automata: Decidable hierarchy problems for stripped-down alternating tree automata. In *CSL*, pages 225–239, 2009.
- 9 E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *29th FOCS*, pages 328–337, 1988.
- 10 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS*, pages 368–377, 1991.
- 11 Alessandro Facchini, Filip Murlak, and Michał Skrzypczak. Index problems for game automata. *ACM Transactions on Computational Logic*, 17(4):24:1–24:38, 2016.
- 12 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
- 13 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65, 1982.
- 14 Christof Löding. Logic and automata over infinite trees. Habilitation thesis, RWTH Aachen University, 2009.
- 15 Andrzej W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Symposium on Computation Theory*, pages 157–168, 1984.
- 16 Damian Niwiński. On fixed-point clones. In *ICALP*, pages 464–473, 1986.
- 17 Damian Niwiński. Fixed point characterization of infinite behavior of finite-state systems. *Theoretical Computer Science*, 189(1–2):1–69, 1997.
- 18 Damian Niwiński and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electronic Notes on Theoretical Computer Science*, 123:195–208, 2005.
- 19 Michael Oser Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.
- 20 Michael Oser Rabin. Weakly definable relations and special automata. In *Proceedings of the Symposium on Mathematical Logic and Foundations of Set Theory*, pages 1–23. North-Holland, 1970.
- 21 Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, pages 389–455. Springer, 1996.
- 22 Klaus Wagner. On ω -regular sets. *Information and Control*, 43(2):123–177, 1979.



Feedback Vertex Set and Even Cycle Transversal for H -Free Graphs: Finding Large Block Graphs

Giacomo Paesani  

Department of Computer Science, Durham University, UK

Daniël Paulusma  

Department of Computer Science, Durham University, UK

Paweł Rzażewski  

Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland

Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw, Poland

Abstract

We prove new complexity results for FEEDBACK VERTEX SET and EVEN CYCLE TRANSVERSAL on H -free graphs, that is, graphs that do not contain some fixed graph H as an induced subgraph. In particular, we prove that both problems are polynomial-time solvable for sP_3 -free graphs for every integer $s \geq 1$; here, the graph sP_3 denotes the disjoint union of s paths on three vertices. Our results show that both problems exhibit the same behaviour on H -free graphs (subject to some open cases). This is in part explained by a new general algorithm we design for finding in a graph G a largest induced subgraph whose blocks belong to some finite class \mathcal{C} of graphs. We also compare our results with the state-of-the-art results for the ODD CYCLE TRANSVERSAL problem, which is known to behave differently on H -free graphs.

2012 ACM Subject Classification Mathematics of computing \rightarrow Graph algorithms

Keywords and phrases Feedback vertex set, even cycle transversal, odd cactus, forest, block

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.82

Related Version *Full Version*: <https://arxiv.org/abs/2105.02736>

Funding *Daniël Paulusma*: Supported by the Leverhulme Trust (RPG-2016-258).

Paweł Rzażewski: Supported by Polish National Science Centre grant no. 2018/31/D/ST6/00062.

Acknowledgements The first author thanks Carl Feghali for an inspiring initial discussion.

1 Introduction

For a set of graphs \mathcal{F} , an \mathcal{F} -transversal of a graph G is a set of vertices that intersects the vertex set of every (not necessarily induced) subgraph of G that is isomorphic to some graph of \mathcal{F} . The problem MIN \mathcal{F} -TRANSVERSAL (also called \mathcal{F} -DELETION) is to find an \mathcal{F} -transversal of minimum size (or size at most k , in the decision variant). Graph transversals form a central topic in Discrete Mathematics and Theoretical Computer Science, both from a structural and an algorithmic point of view.

If \mathcal{F} is the set of all cycles, the set of all even cycles or odd cycles, then we obtain the problems FEEDBACK VERTEX SET, EVEN CYCLE TRANSVERSAL and ODD CYCLE TRANSVERSAL, respectively. All three problems are NP-complete; hence, they have been studied for special graph classes, in particular *hereditary* graph classes, that is, classes closed under vertex deletion. Such classes can be characterized by a (unique) set \mathcal{H} of minimal forbidden induced subgraphs. Then, in order to initiate a systematic study, it is standard to first consider the case where \mathcal{H} has size 1, say $\mathcal{H} = \{H\}$ for some graph H .



© Giacomo Paesani, Daniël Paulusma, and Paweł Rzażewski;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 82; pp. 82:1–82:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We aim to extend known complexity results for FEEDBACK VERTEX SET for H -free graphs and to perform a new, similar study for EVEN CYCLE TRANSVERSAL (for which, so far, mainly parameterized complexity results exist [2, 3, 11, 12]). To describe the known and new results we need some terminology. The cycle and path on r vertices are denoted C_r and P_r , respectively. The *disjoint union* of two vertex-disjoint graphs G_1 and G_2 is the graph $G_1 + G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$. We write sG for the disjoint union of s copies of G . For a set $S \subseteq V$, let $G[S]$ be the subgraph of G induced by S . We write $H \subseteq_i G$ (or $G \supseteq_i H$) if H is an induced subgraph of G .

1.1 Known Results

By Poljak's construction [14], for every integer $g \geq 3$, FEEDBACK VERTEX SET is NP-complete for graphs of girth at least g (the *girth* of a graph is the length of its shortest cycle). The same holds for ODD CYCLE TRANSVERSAL [7]. It is also known that FEEDBACK VERTEX SET [13] and ODD CYCLE TRANSVERSAL [7] are NP-complete for line graphs and thus for claw-free graphs (the claw is the 4-vertex star). Hence, both problems are NP-complete for the class of H -free graphs whenever H has a cycle or claw. A graph with no cycles and no claws is a forest of maximum degree at most 2. Thus, it remains to consider the case where H is a *linear forest*, that is, a collection of disjoint paths. Both problems are polynomial-time solvable on permutation graphs [5] and thus on P_4 -free graphs [5], on sP_2 -free graphs for every $s \geq 1$ [7] and on $(sP_1 + P_3)$ -free graphs for every $s \geq 0$ [9]. Additionally, FEEDBACK VERTEX SET is polynomial-time solvable on P_5 -free graphs [1], and ODD CYCLE TRANSVERSAL is NP-complete for $(P_2 + P_5, P_6)$ -free graphs [9]. A similar NP-hardness result for FEEDBACK VERTEX SET or EVEN CYCLE TRANSVERSAL is unlikely: for every linear forest H , both problems are quasipolynomial-time solvable on H -free graphs [9] (see Section 4 for details).

1.2 Our Results

We first note that MIN \mathcal{F} -TRANSVERSAL is polynomially equivalent to MAX INDUCED \mathcal{F} -SUBGRAPH, the problem of finding a maximum-size induced subgraph of the input graph G that does not belong to \mathcal{F} (where we assume that G has at least one such subgraph). We say that MAX INDUCED \mathcal{F} -SUBGRAPH is the *complementary* problem of MIN \mathcal{F} -TRANSVERSAL, and vice versa. For example, setting $\mathcal{F} = \{P_2\}$ yields the well-known complementary problems MIN VERTEX COVER and MAX INDEPENDENT SET.

Using the complementary perspective, we now argue that FEEDBACK VERTEX SET and EVEN CYCLE TRANSVERSAL are closely related, in contrast to ODD CYCLE TRANSVERSAL. A graph G is *biconnected* if it has at least two vertices, is connected, and $G - u$ is connected for every $u \in V(G)$. A *block* of a graph G is an inclusion-wise maximal biconnected subgraph of G . We now let \mathcal{C} be a set of biconnected graphs. A graph G is a \mathcal{C} -*block graph* if every block of G is isomorphic to some graph in \mathcal{C} . If $\mathcal{C} = \{P_2\}$, then \mathcal{C} -block graphs are precisely forests, and if $\mathcal{C} = \{P_2, C_3, C_5, C_7, \dots\}$, then \mathcal{C} -block graphs are called *odd cacti*. It is well known that a graph is an odd cactus if and only if it does not contain any even cycle as a subgraph. Hence, the complementary problems of EVEN CYCLE TRANSVERSAL and FEEDBACK VERTEX SET are somewhat similar: in particular, both forests and odd cacti have bounded treewidth and their blocks have a very simple structure. This is in stark contrast to ODD CYCLE TRANSVERSAL, whose complementary problem is to find a large induced bipartite subgraph, which might be arbitrarily complicated.

The commonality of complementary problems of EVEN CYCLE TRANSVERSAL and FEEDBACK VERTEX SET leads to the following optimization problem, where \mathcal{C} is some fixed class of biconnected graphs, that is, \mathcal{C} is not part of the input but specified in advance. Note that we consider the more general setting in which every vertex v of G is equipped with a weight $\mathfrak{w}(v) > 0$, and we must find a solution with maximum total weight, respectively.

MAX \mathcal{C} -BLOCK GRAPH

Instance: a graph $G = (V, E)$ with a vertex weight function $\mathfrak{w} : V \rightarrow \mathbb{Q}^+$.

Objective: find a maximum-weight set $X \subseteq V$ such that $G[X]$ is a \mathcal{C} -block graph.

We observe that MAX \mathcal{C} -BLOCK GRAPH is well-defined for every set \mathcal{C} , including $\mathcal{C} = \emptyset$, as every independent set in a graph forms a solution. A restriction of the MAX \mathcal{C} -BLOCK GRAPH problem was introduced and studied from a parameterized complexity perspective by Bonnet et al. [4] as BOUNDED \mathcal{C} -BLOCK VERTEX DELETION (so from the complementary perspective) where each block must in addition have bounded size.

In Section 2 we slightly extend a previously known result, concerning the so-called *blob graphs* [10]. This extended version of the result forms a key ingredient for the proof of our main result, shown in Section 3, which is the following theorem for sP_3 -free graphs (these are the graphs that become a disjoint union of cliques after removing the vertices of any induced $(s-1)P_3$ and their neighbours).

► **Theorem 1.** *For every integer $s \geq 1$ and every finite class \mathcal{C} of biconnected graphs, MAX \mathcal{C} -BLOCK GRAPH can be solved in polynomial time for sP_3 -free graphs.*

Theorem 1 implies the corresponding result for FEEDBACK VERTEX SET, as it is equivalent to MAX $\{P_2\}$ -BLOCK GRAPH. The condition for \mathcal{C} to be finite is critical for our proof technique. Nevertheless, we still have the corresponding result for EVEN CYCLE TRANSVERSAL as well: for sP_3 -free graphs, the cases $\mathcal{C} = \{P_2, C_3, C_5, C_7, \dots\}$ and $\mathcal{C} = \{P_2, C_3, C_5, \dots, C_{4s-3}\}$ are equivalent. Note that we cannot make such an argument for ODD CYCLE TRANSVERSAL, as arbitrarily large bicliques are $2P_3$ -free.

► **Corollary 2.** *For every integer $s \geq 1$, FEEDBACK VERTEX SET and EVEN CYCLE TRANSVERSAL can be solved in polynomial time for sP_3 -free graphs.*

Corollary 2 extends the aforementioned results for FEEDBACK VERTEX SET on sP_2 -free graphs and $(sP_1 + P_3)$ -free graphs. In Section 4 we observe that as a direct consequence of a more general result [1], EVEN CYCLE TRANSVERSAL is polynomial-time solvable for P_5 -free graphs. There we also prove that EVEN CYCLE TRANSVERSAL is NP-complete for graphs of large girth and for line graphs, and consequently, for H -free graphs where H contains a cycle or a claw. Hence, FEEDBACK VERTEX SET and EVEN CYCLE TRANSVERSAL behave similarly on H -free graphs, subject to a number of open cases, which we listed in Table 1.

2 Blob Graph of Graphs With No Large Linear Forest

Let $G = (V, E)$ be a graph. A (*connected*) *component* is a maximal connected subgraph of G . The *neighbourhood* of a vertex $u \in V$ is the set $N_G(u) = \{v \mid uv \in E\}$. For $U \subseteq V$, we let $N_G(U) = \bigcup_{u \in U} N(u) \setminus U$. Two sets $X_1, X_2 \subseteq V(G)$ are *adjacent* if $X_1 \cap X_2 \neq \emptyset$ or there exists an edge with one endpoint in X_1 and the other in X_2 . The *blob graph* G° of G is defined as follows.

$$V(G^\circ) := \{X \subseteq V(G) \mid G[X] \text{ is connected}\} \text{ and } E(G^\circ) := \{X_1 X_2 \mid X_1 \text{ and } X_2 \text{ are adjacent}\}.$$

Gartland et al. [10] showed that for every graph G , the length of a longest induced path in G° is equal to the length of a longest induced path in G . We slightly generalize this result.

■ **Table 1** The complexity of FEEDBACK VERTEX SET (FVS), EVEN CYCLE TRANSVERSAL (ECT) and ODD CYCLE TRANSVERSAL (OCT) on H -free graphs for a linear forest H . All three problems are NP-complete for H -free graphs when H is not a linear forest (see also Section 4). The two blue cases (one for FVS, one for ECT) are the *algorithmic* contributions of this paper. We write $H \subseteq_i H'$ if H is an induced subgraph of H' . See Section 1.1 for references to the known results in the table.

	polynomial-time	unresolved	NP-complete
FVS	$H \subseteq_i P_5$ or sP_3 for $s \geq 1$	$H \supseteq_i P_1 + P_4$	none
ECT	$H \subseteq_i P_5$ or sP_3 for $s \geq 1$	$H \supseteq_i P_1 + P_4$	none
OCT	$H = P_4$ or $H \subseteq_i sP_1 + P_3$ or sP_2 for $s \geq 1$	$H = sP_1 + P_5$ for $s \geq 0$ or $H = sP_1 + tP_2 + uP_3 + vP_4$ for $s, t, u \geq 0, v \geq 1$ with $\min\{s, t, u\} \geq 1$ if $v = 1$, or $H = sP_1 + tP_2 + uP_3$ for $s, t \geq 0,$ $u \geq 1$ with $u \geq 2$ if $t = 0$	$H \supseteq_i P_6$ or $P_2 + P_5$

► **Theorem 3.** *For every linear forest H , a graph G contains H as an induced subgraph if and only if G° contains H as an induced subgraph.*

Proof. As G is an induced subgraph of G° , the (\Rightarrow) implication is immediate. We prove the (\Leftarrow) implication by induction on the number k of connected components of H . If $k = 1$, then the claim follows directly from the aforementioned result of Gartland et al. [10]. So assume that $k \geq 2$ and the statement holds for all linear forests H with fewer than k connected components. Let P' be one of the connected components of H , and define $H' := H - P'$.

Suppose that G° contains an induced subgraph isomorphic to H . Let \mathcal{X} be the set of vertices of G° , such that $G^\circ[\mathcal{X}]$ is isomorphic to H . Furthermore, let $\mathcal{Y} \subseteq \mathcal{X}$ be the set of vertices that induce in $G^\circ[\mathcal{X}]$ the component P' of H , that is, $G^\circ[\mathcal{Y}]$ is isomorphic to P' .

Let $Y \subseteq V(G)$ be the union of sets in \mathcal{Y} . Note that $G^\circ[\mathcal{Y}]$ is an induced subgraph of $(G[Y])^\circ$. Thus, by the inductive assumption, $G[Y]$ contains an induced copy of P' .

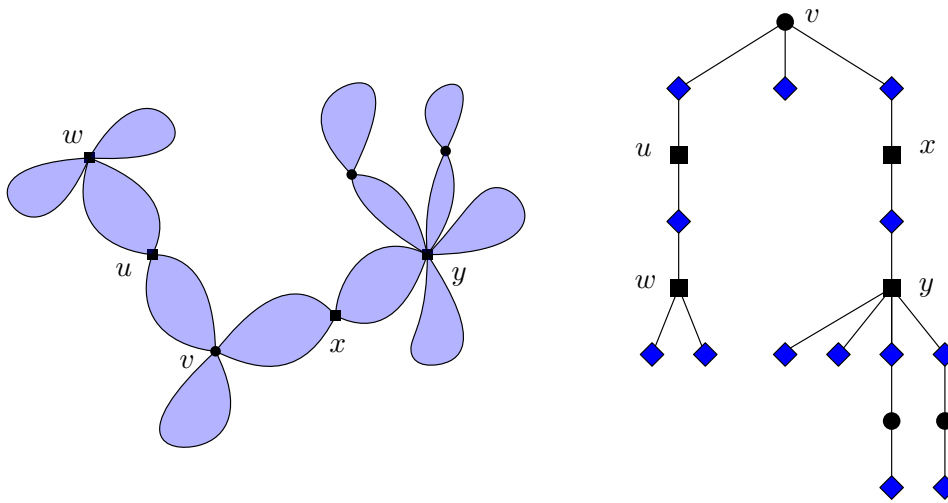
Let $X \subseteq V(G)$ be the union of sets in $\mathcal{X} \setminus \mathcal{Y}$. Since the copy of H in G° is induced, we know that in G° there are no edges between $\mathcal{X} \setminus \mathcal{Y}$ and \mathcal{Y} . This is equivalent to saying that $X \cap N[Y] = \emptyset$. So we conclude that $G^\circ[\mathcal{X} \setminus \mathcal{Y}]$ is an induced subgraph of $(G - N[Y])^\circ$. Since $G^\circ[\mathcal{X} \setminus \mathcal{Y}]$, and thus $(G - N[Y])^\circ$, contains an induced copy of H' , by the inductive assumption we know that $G - N[Y]$ contains an induced copy of H' . Combining this subgraph with the induced copy of P' in $G[Y]$, we obtain an induced copy of H in G . ◀

3 The Proof of Theorem 1

We start with analyzing the structure of sP_3 -free \mathcal{C} -block graphs in Section 3.1, where \mathcal{C} is any finite class of biconnected graphs. Then, in Section 3.2, we present our algorithm for MAX \mathcal{C} -BLOCK GRAPH on sP_3 -free graphs.

3.1 Blocks and Terminals in sP_3 -free Graphs

From now on, let \mathcal{C} be a finite class of biconnected graphs. For some fixed positive integer s , let $G = (V, E)$ be an sP_3 -free graph with n vertices and vertex weights $\mathbf{w}: V \rightarrow \mathbb{Q}^+$. Let $X \subseteq V$ such that $F = G[X]$ is a \mathcal{C} -block graph. A component of F is *trivial* if it is a single vertex or a single block, otherwise it is *non-trivial*. Let F' be the graph obtained from F by removing all trivial components. Note that F' and F are sP_3 -free, as G is sP_3 -free.



■ **Figure 1** Left: a graph F' . Blue shapes are blocks, squares are terminals, and dots are non-terminal cutvertices. Right: $\text{BCF}(F')$, rooted in the cutvertex v . Blue diamonds are blocks; w is a terminal of type 1, u and x are terminals of type 2, and y is a terminal of both types. The remaining cutvertices are not terminals. We also use this example with this particular $\text{BCF}(F')$ in later figures.

We denote the set of cutvertices of F' and the set of blocks of F' by $\text{Cutvertices}(F')$ and $\text{Blocks}(F')$, respectively. The *block-cut forest* $\text{BCF}(F')$ of F' has vertex set $\text{Cutvertices}(F') \cup \text{Blocks}(F')$ and an edge set that consists of all edges xb such that $x \in \text{Cutvertices}(F')$ and $b \in \text{Blocks}(F')$, and x belongs to b . By definition, each component of F' has a cutvertex; we pick an arbitrary one as root for the corresponding tree in $\text{BCF}(F')$ to get a parent-child relation. Each leaf of $\text{BCF}(F')$ belongs to $\text{Blocks}(F')$, and we call such blocks *leaf blocks*.

A cutvertex x of F' is a *terminal of type 1* if x has at least two children in $\text{BCF}(F')$ that are leaves, whereas x is a *terminal of type 2* if there exists a leaf block, whose great-grandparent in $\text{BCF}(F')$ is x . In the latter case, there is a three-edge downward path from x to a leaf in $\text{BCF}(F')$; see also Fig. 1. Let d be the maximum number of vertices of a graph in \mathcal{C} .

► **Lemma 4.** *At most $d \cdot (s - 1)$ vertices of F' are terminals of type 1.*

Proof. For contradiction, suppose that there are at least $d \cdot (s - 1) + 1$ terminals of type 1. We observe that F' is d -colourable. Indeed, each block has at most d vertices, so d colours are sufficient to colour each block. Furthermore, we can permute the colours in each block, so that the colourings agree on cutvertices.

This implies that there is an independent set X of size at least s , whose every element is a terminal of type 1. For each such terminal v , let its *private* P_3 be a 3-vertex path with v as the central vertex and each endpoint belonging to a different leaf block that is a child of v in $\text{BCF}(F')$. Note that each private P_3 is induced. Furthermore, the private P_3 's of vertices in X are pairwise non-adjacent: this follows from the definition of terminals of type 1 and the fact that X is independent. Thus we have found an induced sP_3 in F , a contradiction. ◀

► **Lemma 5.** *At most $(d + 1) \cdot (s - 1)$ vertices of F' are terminals of type 2.*

Proof. For contradiction, suppose that there are at least $(d + 1) \cdot (s - 1) + 1$ terminals of type 2. Observe that F' has a proper $(d + 1)$ -colouring f , satisfying the following two properties:

1. the vertices in each block receive pairwise distinct colours and
 2. if b is a block, then any vertex of b receives a colour which is different than the colour of the cutvertex which is the great-grandparent of b in $\text{BCF}(F')$ (if such a cutvertex exists).
- It is easy to find such a colouring of each tree in $\text{BCF}(F')$ by choosing an arbitrary colour for the root and proceeding in a top-down fashion. Suppose we want to colour the block b and its parent in $\text{BCF}(F')$ is the cutvertex v . Recall that b has at most d vertices and exactly one of them is already coloured. Furthermore, we want to avoid the colour of the grandparent of v (if such a vertex exists), so we have sufficiently many free colours to colour each vertex of $b \setminus \{v\}$ with a different one.

Now, by our assumption, there is a set X of at least s terminals of type 2 that received the same colour in f . For each $v \in X$, we define its *private* P_3 as follows. Recall that by the definition of a terminal of type 2, there is a leaf block b , whose great-grandparent in $\text{BCF}(F')$ is v . The private P_3 of v is given by the first three vertices on a shortest path P from v to b . Note that in the extreme case it might happen that both b and its grandparent in $\text{BCF}(F')$ are edges, but P always has at least three vertices.

Clearly, each private P_3 is an induced path. We claim that the private P_3 's associated with the vertices of X are non-adjacent. For contradiction, suppose otherwise. Let v_1, v_2 be distinct vertices of X , and let v_i, x_i, y_i be the consecutive vertices of the private P_3 associated with v_i . Let b_i be the block containing v_i and x_i .

First, observe that the sets $\{v_1, x_1, y_1\}$ and $\{v_2, x_2, y_2\}$ are disjoint. Indeed, we know that $v_1 \neq v_2$ and because $\text{BCF}(F')$ is a rooted tree, we have that $\{x_1, y_1\} \cap \{x_2, y_2\} = \emptyset$. Furthermore, recall that $f(v_1) = f(v_2)$ and by the definition of f , we have that the colours of x_i and of y_i are different from the colour of v_i .

So now suppose that there is an edge with one endpoint in $\{v_1, x_1, y_1\}$ and the other in $\{v_2, x_2, y_2\}$. Clearly this edge cannot join v_1 and v_2 , as the colouring f is proper. Furthermore, there is no edge between $\{x_1, y_1\}$ and $\{x_2, y_2\}$, as v_1 and v_2 are cutvertices of a rooted tree. Suppose that v_2 is adjacent to x_1 (the case that v_1 is adjacent to x_2 is symmetric). As each vertex of b_1 gets a different colour in f , we observe that v_2 cannot belong to b_1 . Thus x_1 is a cutvertex. However, by the second property of f , we obtain that the colour of v_2 must be different than the colour of v_1 , a contradiction.

So finally suppose that v_2 is adjacent to y_1 . Note that then y_1 cannot belong to a leaf block, meaning that y_1 belongs to b_1 . Similarly to the previous paragraph, the definition of f implies that the colour of v_2 must be different than the colour of v_1 , a contradiction.

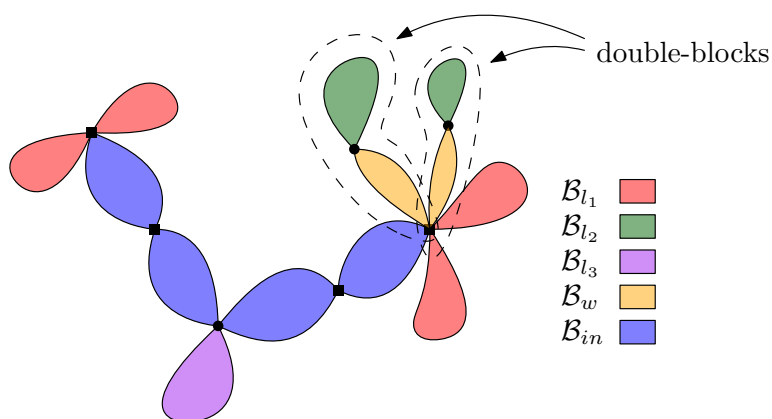
Thus we have found an induced sP_3 in F' , a contradiction. \blacktriangleleft

Lemmas 4 and 5 imply the following.

► **Lemma 6.** *The number of terminals of F' is at most $(2d + 1) \cdot (s - 1)$.*

If v is a terminal of type 2, then by definition there is a cutvertex w that belongs to both a block containing v as well as to some leaf block. We call such w a *witness* of v . We now partition the set of blocks of F' into the following subsets; see also Fig. 2:

- \mathcal{B}_{i_1} is the set of leaf blocks containing a terminal of type 1,
- \mathcal{B}_{i_2} is the set of leaf blocks containing a witness w that is not a terminal of type 1,
- \mathcal{B}_{i_3} is the set of remaining leaf blocks, that is, the ones with a cutvertex that is neither a terminal nor a witness,
- \mathcal{B}_w is the set of blocks with precisely two cutvertices, one of which is a terminal of type 2 and the other one the non-terminal witness of that type-2 terminal, and
- \mathcal{B}_{in} is the set of all remaining blocks.



■ **Figure 2** The classification of blocks of the example of Figure 1.

Note that blocks in \mathcal{B}_{t_2} and \mathcal{B}_w come in pairs, that is, for each block B in one of these sets, there is exactly one block B' in the other set, such that B and B' share a vertex (it is the witness w , using the notation introduced above). We will consider these two blocks as one object. Formally, a *double-block* is a graph consisting of two blocks sharing a cutvertex. Let \mathcal{B}_d be the family of double-blocks of F' obtained from blocks in \mathcal{B}_{t_2} and \mathcal{B}_w in the way described above, i.e., \mathcal{B}_d consists of graphs $G[V(B) \cup V(B')]$, where $B \in \mathcal{B}_{t_2}$, $B' \in \mathcal{B}_w$ and $V(B) \cap V(B') \neq \emptyset$. Note that each double-block in \mathcal{B}_d has fewer than $2d$ vertices and contains exactly one terminal of type 2.

A *backbone* of a component Z of F' is a minimum tree T_Z contained in Z that connects all terminals of F' that belong to Z ; observe that all leaves of T_Z are terminals. The *skeleton* S of F' is the graph obtained from F' by removing all vertices from the blocks in \mathcal{B}_{t_1} except terminals of type 1 and all vertices from the double-blocks in \mathcal{B}_d except terminals of type 2. Note that every backbone is a subgraph of S .

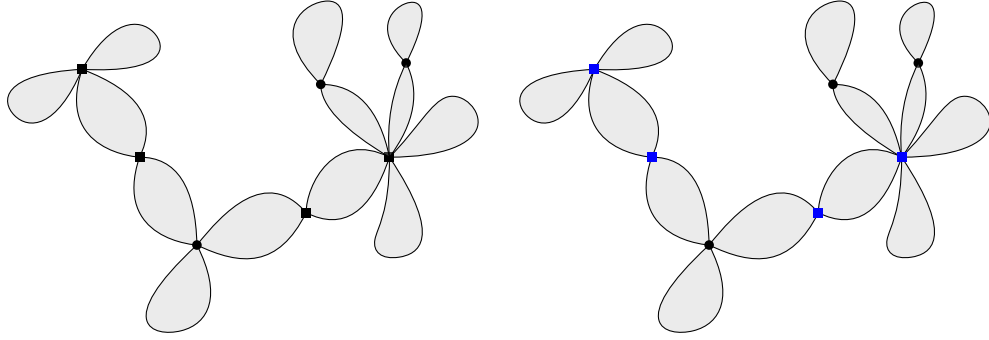
3.2 The Algorithm

Outline. Our polynomial-time algorithm consists of the following two phases:

1. *Branching Phase*, which consists of the following three steps:
 1. guessing the terminals of F' ;
 2. guessing the backbones of the components of F' ; and
 3. guessing the skeleton of F' , and
2. *Completion Phase*, where we extend the partial solutions obtained in the Branching Phase to complete ones by finding non-skeleton vertices of F' and trivial components of F' ; we do this by:
 1. reducing the problem to MAX WEIGHT INDEPENDENT SET for sP_3 -free graphs using the blob graph construction in Section 2, and
 2. solving this problem using the polynomial-time algorithm of Brandstädt and Mosca [6].

We now describe our algorithm, prove its correctness and perform a running time analysis.

Branching Phase. This phase of our algorithm consists of a series of guesses, where we find certain vertices and substructures in G . The total number of vertices to be guessed will be $\mathcal{O}(s^2 d^2)$. Since we guess them exhaustively, this results in a recursion tree with $\mathcal{O}(n^{\mathcal{O}(s^2 d^2)})$ leaves. As both s and d are constants, this bound is polynomial in n . We will ensure that



■ **Figure 3** Step 1 of the Branching Phase. Left: the graph F' . Right: the terminals of F' .

the optimum solution $F = G[X]$ will be found in the call corresponding to at least one of the leaves of the recursion tree. Based on the properties of F , we will expect the guessed vertices to satisfy certain conditions. If, at some point, the guessed vertices do not satisfy these conditions, we just terminate the current call, as it will not lead us to find F . This will be applied implicitly throughout the execution of the algorithm.

The branching phase is illustrated in Figures 3–5. We use the convention that gray/black elements are still unknown and blue elements are the ones that we have already guessed.

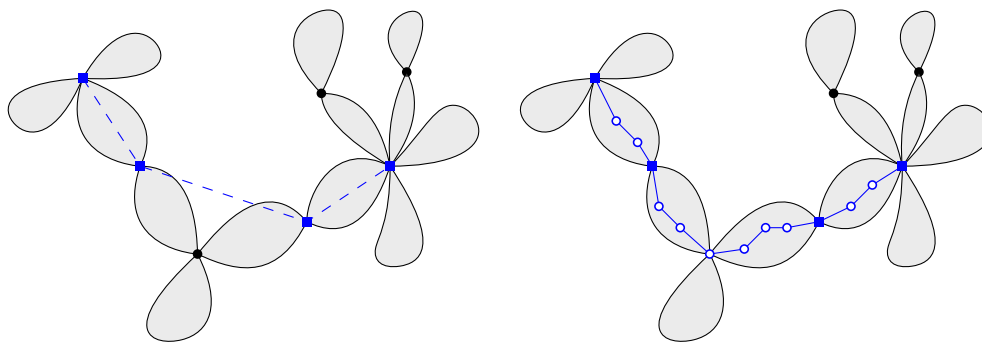
Step 1. Guessing the terminals of F' . We guess the set $C \subseteq V$ of terminals of F' . By Lemma 6, the total number of terminals is bounded by $(2d + 1) \cdot (s - 1) \leq 3ds$. Furthermore, for each terminal, we guess its type (1, 2, or both). This results in $3^{|C|} \leq 3^{3ds}$ possibilities. We also guess the partition of C , corresponding to the connected components of F . This results in at most $|C|^{|C|} \leq (3ds)^{3ds}$ additional branches. In total, we have $\mathcal{O}(n^{\mathcal{O}(ds)})$ branches.

Step 2. Guessing the backbone of each component of F' . Let Z be a component of F' . Let $C_Z \subseteq C$ be the subset of terminals that are in Z . Let T_Z be the backbone of Z . Let T'_Z be the tree obtained from T_Z by contracting every path in T_Z whose internal vertices are all non-terminals and of degree 2 to an edge. Note that every non-terminal vertex of T'_Z has degree at least 3. Since T'_Z has at most $|C_Z|$ vertices of degree at most 2, by the handshaking lemma we observe that the total number of vertices of T'_Z is at most $2|C_Z|$. Recall that every edge of T'_Z corresponds to an induced path in T_Z . Since F' is sP_3 -free and thus P_{4s-1} -free, we conclude that T_Z has at most $2|C_Z| \cdot (4s - 2) \leq 8s \cdot |C_Z|$ vertices.

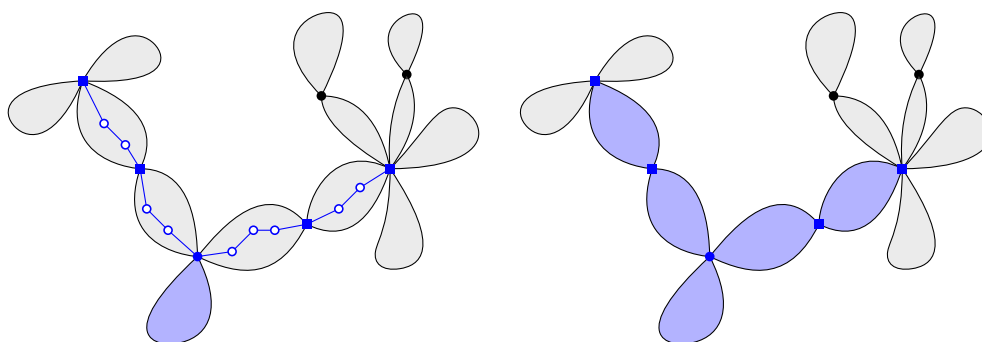
Let T be the forest whose components are the guessed backbones of the components of F' . Note that the total number of vertices of T is at most $\sum_Z 8s \cdot |C_Z| = 8s \cdot |C| \leq 24ds^2$. Thus we may guess the whole forest T , which results in $\mathcal{O}(n^{\mathcal{O}(ds^2)})$ branches.

Step 3. Guessing the skeleton of F' . Let T be the forest guessed in the previous step; recall that T has at most $24ds^2$ vertices. We guess the partition of $E(T)$ corresponding to *blocks* of F' ; note that a vertex v may be in several blocks: this happens precisely if v is a cutvertex in F' . This results in at most $|E(T)|^{\mathcal{O}(|E(T)|)} \leq |V(T)|^{\mathcal{O}(|V(T)|)} \leq (ds)^{\mathcal{O}(ds^2)}$ branches.

We now discuss some properties of the (double-)blocks. We use the names of vertices as in the definitions introduced in Section 3.1, recall also Fig. 2. The crucial observation is that now there is a branch, where:



■ **Figure 4** Step 2 of the Branching Phase. Left: the tree T'_Z . Right: the tree T_Z .



■ **Figure 5** Step 3 of the Branching Phase. Left: our knowledge about F' after guessing the blocks in \mathcal{B}_{l_3} . Right: our knowledge about F' after guessing the blocks in \mathcal{B}_{in} .

- For each block in \mathcal{B}_{l_1} , we have guessed its cutvertex and no other vertices.
- For each block in \mathcal{B}_{l_2} , we have not guessed any vertices.
- For each block in \mathcal{B}_{l_3} , we have guessed its cutvertex v connecting it to the rest of F' and no other vertices; note that v is not a terminal. Moreover, for each such v there is at most one block in \mathcal{B}_{l_3} .
- For each block in \mathcal{B}_w , we have guessed its cutvertex v that does not belong to a block in \mathcal{B}_{l_2} and we guessed no other vertices. Thus, for each double-block in \mathcal{B}_d , we have guessed its cutvertex connecting it to the rest of F' and no other vertices.
- For each block in \mathcal{B}_{in} , we have guessed at least two vertices.

Now we proceed to the final guessing step, see Fig. 5. First, we guess all blocks in \mathcal{B}_{l_3} . Note that we can do it, as (i) we know their cutvertices, (ii) the number of these cutvertices is at most $|V(T)| \leq 24ds^2$, (iii) each cutvertex is contained in at most one block from \mathcal{B}_{l_3} , and (iv) each block has at most d vertices. This results in at most $n^{\mathcal{O}(|V(T)| \cdot d)} = n^{\mathcal{O}(d^2 s^2)}$ branches.

Next, we guess all blocks in \mathcal{B}_{in} . Again, we can do it as (i) we know at least two vertices of such a block, (ii) the number of these blocks is at most $|E(T)| \leq 24ds^2$, and (iii) each block has at most d vertices. This results in at most $n^{\mathcal{O}(|V(T)| \cdot d)} = n^{\mathcal{O}(d^2 s^2)}$ further branches.

The following claim summarizes the outcome of the guessing phase of the algorithm.

▷ **Claim A.** In time $\mathcal{O}(n^{\mathcal{O}(s^2 d^2)})$ we can enumerate a collection \mathcal{S} of $\mathcal{O}(n^{\mathcal{O}(s^2 d^2)})$ triples (S, C_1, C_2) , where $S \subseteq V$ and $C_1, C_2 \subseteq S$ such that \mathcal{S} has the following property. Let $X \subseteq V$, such that $F = G[X]$ is a \mathcal{C} -block graph. Let $X' \subseteq X$ be the vertex set of the graph F' obtained from F by removing all trivial components. Then there is at least one triple $(S, C_1, C_2) \in \mathcal{S}$, where

82:10 Feedback Vertex Set and Even Cycle Transversal for H -Free Graphs

- a) C_1 is the set of terminals of type 1 in F' ,
- b) C_2 is the set of terminals of type 2 in F' ,
- c) $G[S]$ is the skeleton of F' .

Completion Phase. Let \mathcal{S} be the collection from Claim A and let $(S, C_1, C_2) \in \mathcal{S}$ be a triple that satisfies the properties listed in the statement of Claim A for an optimum solution $F = G[X]$. Let $\mathcal{X} := \mathcal{X}_0 \cup \mathcal{X}_1 \cup \mathcal{X}_2$ be the family of subsets of V with:

$$\begin{aligned}\mathcal{X}_0 &:= \{\{v\} \mid v \in V\}, \\ \mathcal{X}_1 &:= \{B \subseteq V \mid G[B] \in \mathcal{C}\}, \text{ and} \\ \mathcal{X}_2 &:= \{B \subseteq V \mid B \text{ is a double-block whose blocks are in } \mathcal{C}\},\end{aligned}$$

Let $G^{\mathcal{C}}$ be the graph whose vertex set is \mathcal{X} , and edges join sets that are adjacent in G . Furthermore, we define a weight function $\mathfrak{w}^{\mathcal{C}}: \mathcal{X} \rightarrow \mathbb{Q}^+$ as

$$\mathfrak{w}^{\mathcal{C}}(A) = \sum_{v \in A} \mathfrak{w}(v).$$

Note that in order to complete S to the optimum solution $F = G[X]$, we need to determine:

- all blocks in \mathcal{B}_{l_1} ,
- all double-blocks in \mathcal{B}_d ,
- all trivial components of F .

Note that the vertex sets of all these subgraphs are in the family \mathcal{X} and they form an independent set in $G^{\mathcal{C}}$. Furthermore, since X is of maximum weight, the total weight of selected subsets must be maximized. Thus the idea behind the last step is to reduce the problem to solving MAX WEIGHT INDEPENDENT SET in an appropriately defined subgraph of $G^{\mathcal{C}}$ and weights $\mathfrak{w}^{\mathcal{C}}$.

To ensure that the selected subsets are consistent with our guess $(S, C_1, C_2) \in \mathcal{S}$, we will remove certain vertices from $G^{\mathcal{C}}$. In particular, let \mathcal{X}' consist of the sets $A \in \mathcal{X}$, such that:

1. $A \in \mathcal{X}_0 \cup \mathcal{X}_1$ and A is non-adjacent to S ; these are the candidates for trivial components of F ,
2. $A \in \mathcal{X}_1$ and A intersects S in exactly one vertex, which is in C_1 ; these are the candidates for blocks in \mathcal{B}_{l_1} ,
3. $A \in \mathcal{X}_2$ and A intersects S in exactly one vertex, which is in C_2 and is not the cutvertex of $G[A]$; these are the candidates for double-blocks in \mathcal{B}_d .

Now let $\mathcal{I} \subseteq \mathcal{X}'$ be an independent set of $G^{\mathcal{C}}$, and let $S' = \bigcup_{A \in \mathcal{I}} A$. It is straightforward to verify that if $(S, C_1, C_2) \in \mathcal{S}$ satisfies the properties listed in Claim A, then $G[S \cup S']$ is a \mathcal{C} -block graph. Thus, in one of the branches, we will find the optimum solution $F = G[X]$.

Now let us argue that the last step can be performed in polynomial time. First, observe that $|\mathcal{X}| \leq n + n^d + n^{2d} = n^{\mathcal{O}(d)}$ and the family \mathcal{X} can be exhaustively enumerated in time $n^{\mathcal{O}(d)}$. Next, \mathcal{X}' can be computed in time polynomial in $|\mathcal{X}|$, and thus in n . This implies that the graph $G^{\mathcal{C}}[\mathcal{X}']$ can be computed in time polynomial in n . We observe that $G^{\mathcal{C}}$, and thus $G^{\mathcal{C}}[\mathcal{X}']$, is an induced subgraph of the blob graph G° , introduced in Section 2. Hence, by Theorem 3, we conclude that $G^{\mathcal{C}}[\mathcal{X}']$ is sP_3 -free.

The final ingredient is the polynomial-time algorithm for MAX WEIGHT INDEPENDENT SET in sP_3 -free graphs by Brandstädt and Mosca [6]. Its running time on an n' -vertex graph is $n'^{\mathcal{O}(s)}$. Since the number of vertices of $G^{\mathcal{C}}[\mathcal{X}']$ is $n^{\mathcal{O}(d)}$, we conclude that a maximum-weight independent set in $G^{\mathcal{C}}[\mathcal{X}']$ can be found in time $n^{\mathcal{O}(sd)}$.

Summing up, in the guessing phase, in time $n^{\mathcal{O}(s^2 d^2)}$ we enumerate the family \mathcal{S} of size $n^{\mathcal{O}(s^2 d^2)}$. Then, for each member (S, C_1, C_2) of \mathcal{S} , we try to extend the partial solution to a complete one. This takes time $n^{\mathcal{O}(sd)}$ per element of \mathcal{S} . Among all found solutions, we return the one with maximum weight. The total running time of the algorithm is $n^{\mathcal{O}(s^2 d^2)}$, which is polynomial in n , since s and d are constants. This completes the proof of Theorem 1.

4 More Results for Even Cycle Transversal on H -Free Graphs

In this section we prove that subject to a number of unsolved cases, the complexity of EVEN CYCLE TRANSVERSAL for H -free graphs coincides with the one for FEEDBACK VERTEX SET.

CMSO₂ and Even Cycle Transversal. *Monadic Second-Order Logic* (MSO₂) over graphs consists of formulas with vertex variables, edge variables, vertex set variables, and edge set variables, quantifiers, and standard logic operators. We also have a predicate $\text{inc}(v, e)$, indicating that the vertex v belongs to the edge e . *Counting Monadic Second-Order Logic* (CMSO₂) is an extension of MSO₂ which allows atomic formulas of the form $|X| \equiv p \pmod q$, where X is a set variable and $0 \leq p < q$ are integers.

Abrishami et al. [1, Theorems 5.3 and 7.3] proved that for any fixed CMSO₂ formula Φ and any constant t , the following problem is polynomial-time solvable: given a P_5 -free graph G with weight function $\mathfrak{w} : V(G) \rightarrow \mathbb{Q}^+$, find a maximum-weight set $X \subseteq V(G)$, such that $G[X]$ is of treewidth at most t and satisfies Φ . This immediately yields a polynomial-time algorithm for FEEDBACK VERTEX SET in P_5 -free graphs: just take $t = 1$ and a trivial formula Φ that is satisfied for all graphs (see also [1]).

A similar argument can also be applied for EVEN CYCLE TRANSVERSAL. First, note that every odd cactus has treewidth at most 2. Hence, it remains to show an appropriate CMSO₂ formula Φ that enforces $G[X]$ to be an odd cactus. We will again look from the complementary perspective: we need to say that $G[X]$ has no even cycle. For this, it is enough to say that there is no set E' of edges in $G[X]$, such that: (i) each vertex of X is incident to 0 or 2 edges from E' , (ii) the edges from E' induce a connected subgraph of $G[X]$, and (iii) the number of edges in E' is even. Properties (i) and (ii) are easily expressible in MSO₂, see [8, Section 7.4], and property (iii) is expressed by the formula $|E'| \equiv 0 \pmod 2$, which is allowed in CMSO₂. This immediately yields the following corollary.

► **Corollary 7.** EVEN CYCLE TRANSVERSAL is polynomial-time solvable for P_5 -free graphs.

Finally, the problem of finding a maximum-weight subset that induces a constant-treewidth graph satisfying some fixed CMSO₂ formula can be solved in *quasipolynomial time* for P_r -free graphs for any fixed r [10]. This implies a quasipolynomial-time algorithm for FEEDBACK VERTEX SET and EVEN CYCLE TRANSVERSAL for H -free graphs if H is a linear forest.

Hardness Results. An *odd cycle factor* of a graph G is a set of odd cycles such that every vertex of G belongs to exactly one of them. The ODD CYCLE FACTOR problem, which asks if a graph has an odd cycle factor, is known to be NP-complete [15]. The *line graph* $L(G)$ of a graph $G = (V, E)$ has vertex set E and an edge between two distinct vertices e and f if and only if e and f share an end-vertex in G .

The proof of our next result for line graphs is somewhat similar to a proof for ODD CYCLE TRANSVERSAL of [7] but uses some different arguments as well.

► **Theorem 8.** EVEN CYCLE TRANSVERSAL is NP-complete for line graphs.

82:12 Feedback Vertex Set and Even Cycle Transversal for H -Free Graphs

Proof. Let $G = (V, E)$ be an instance of ODD CYCLE FACTOR with n vertices and m edges. We claim that G has an odd cycle factor if and only if its line graph $L := L(G)$ has an even cycle transversal of size at most $m - n$, see Fig. 6.

First suppose G has an odd cycle factor. Then there is $E' \subseteq E$, such that $|E'| = n$ and $L[E']$ is a disjoint union of odd cycles. Hence, $S := E \setminus E'$ is an even cycle transversal of L of size $|E| - n = m - n$. Now suppose L has an even cycle transversal S with $|S| \leq m - n$. Let $E' := E \setminus S$. As $|E| = m$, we have $|E'| \geq n$.

We prove the following claim.

▷ **Claim B.** Every component of $L[E']$ is either an odd cycle or the line graph of a tree.

Proof. Let D be a component of $L[E']$. If D has no cycle, then D is a path, as L is a line graph and thus is claw-free. Hence, D is the line graph of a path, and thus a tree.

So suppose D has a cycle C . Then C is odd and induced, as $L[E']$ is an odd cactus. If D has no vertices except for the ones of C , then D is an odd cycle and we are done. Suppose otherwise.

First, assume that C has at least five vertices. Since D has vertices outside C , there is a vertex of C with a neighbour outside C . Hence, D contains either an even cycle or an induced claw, both of which are not possible. So now suppose that C has at most four vertices. Then C is a triangle, as D has no even cycles. Since D is an induced subgraph of L , there exists a subgraph T of G such that $D = L(T)$. As D is a connected graph with at least four vertices, containing a triangle, T is a connected graph with at least four vertices.

We aim to show that T is a tree. For contradiction, suppose that T contains a cycle C_T . Then C_T must be a triangle, as otherwise D would contain an even cycle or an odd cycle with at least five vertices. Let a, b, c be the vertices of C_T . As T is connected and has at least four vertices, at least one of $\{a, b, c\}$, say a , must have a neighbour $d \notin \{b, c\}$. However, the edges $ad - ab - bc - ac$ form a C_4 in D , a contradiction with D being an odd cactus. So we conclude that T contains no cycles and thus T is a tree. ◁

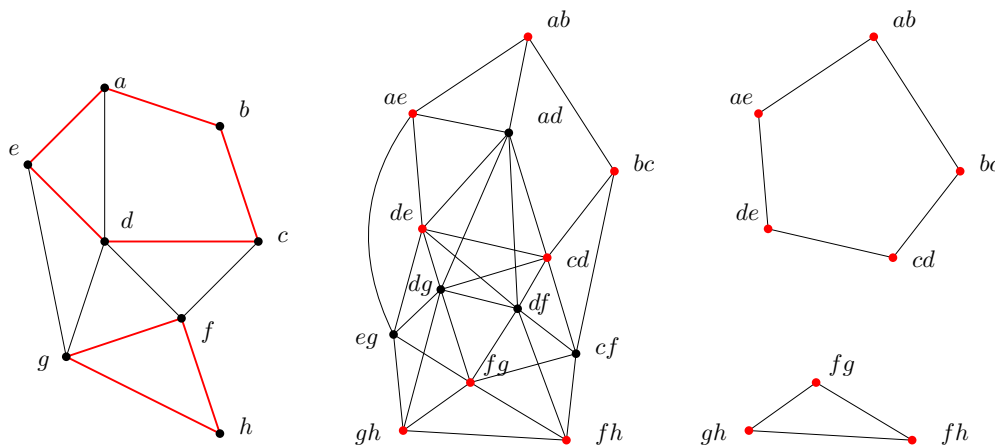
Each component of $L[E']$ that is an odd cycle corresponds to an odd cycle in G . By Claim B, each component D of $L[E']$ that is not an odd cycle is the line graph of some subtree T of G . So, if D has r vertices, then T has $r + 1$ vertices. Furthermore, the vertex sets of G corresponding to distinct components of $L[E']$ are pairwise disjoint. Suppose that $L[E']$ has $p \geq 0$ components that are not odd cycles. Let Q be the set of vertices incident to at least one edge of E' . Then $n = |V(G)| \geq |Q| = |E'| + p \geq n + p$. Hence, $p = 0$ and $|Q| = n$. So, the components of $L[E']$ correspond to an odd cycle factor of G . This completes the proof. ◀

We make a straightforward observation similar to an observation for FEEDBACK VERTEX SET [7, 14], except that we must subdivide edges of a graph an even number of times.

► **Theorem 9.** For every $p \geq 3$, EVEN CYCLE TRANSVERSAL is NP-complete for graphs of girth at least p .

Proof. We reduce from EVEN CYCLE TRANSVERSAL for general graphs by noting the following. Namely, the size of a minimum even cycle transversal in G is equal to the size of a minimum even cycle transversal in the graph G' obtained from G by subdividing every edge $2p$ times, and the girth of G' is at least p . ◀

The next theorem is analogous to the one for FEEDBACK VERTEX SET; see also Table 1.



■ **Figure 6** Left: a graph G with an odd cycle factor. Middle: the graph $L = L(G)$ and the set E' (red). Black vertices form an even cycle factor. Right: the odd cactus $L[E']$.

► **Theorem 10.** *Let H be a graph. Then EVEN CYCLE TRANSVERSAL for H -free graphs is polynomial-time solvable if $H \subseteq_i sP_3$ for some $s \geq 1$ or $H \subseteq_i P_5$, and it is NP-complete if H is not a linear forest.*

Proof. If $H \subseteq_i sP_3$, use Corollary 2, and if $H \subseteq_i P_5$, use Corollary 7. If H is not a linear forest, then it has a cycle or a claw. If H has a cycle, then we apply Theorem 9 for $p = |V(H)| + 1$. Otherwise, H has an induced claw and we apply Theorem 8. ◀

5 Conclusions

We prove that for a large family of graphs \mathcal{F} , the MIN \mathcal{F} -TRANSVERSAL problem is polynomial-time solvable on sP_3 -free graphs (for every $s \geq 1$). The two best-known problems in this framework are FEEDBACK VERTEX SET and EVEN CYCLE TRANSVERSAL. Our result for FEEDBACK VERTEX SET generalizes two known results from the literature [7, 9]. We also prove that in contrast to the situation for ODD CYCLE TRANSVERSAL, all other known complexity results for FEEDBACK VERTEX SET on H -free graphs hold for EVEN CYCLE TRANSVERSAL as well. Hence, so far both problems behave the same on special graph classes, and it would be interesting to prove polynomial equivalency of the two problems more generally. Table 1 still shows some missing cases for each of the three problems.

In particular, we highlight a borderline case:

Is each of the three problems is polynomial-time solvable for $(P_1 + P_4)$ -free graphs?

The main obstacle is that we know no polynomial-time algorithm for finding a maximum induced disjoint union of stars in a $(P_1 + P_4)$ -free graph; note that such a subgraph could be a potential optimal solution for each of the three problems.

We also recall that FEEDBACK VERTEX SET and EVEN CYCLE TRANSVERSAL can be solved in quasipolynomial time for P_r -free graphs [10] for every $r \geq 1$, whereas ODD CYCLE TRANSVERSAL is NP-complete even for P_6 -free graphs [9]. An affirmative answer to the above question for FEEDBACK VERTEX SET and EVEN CYCLE TRANSVERSAL would be a first step in proving that these two problems are polynomial-time solvable on P_6 -free graphs. If that turns out to be the case, then we will have further evidence that these two problems, restricted to H -free graphs, differ in their complexity from ODD CYCLE TRANSVERSAL.

References

- 1 Tara Abrishami, Maria Chudnovsky, Marcin Pilipczuk, Paweł Rzażewski, and Paul Seymour. Induced subgraphs of bounded treewidth and the container method. *Proc. SODA 2021*, pages 1948–1964, 2021.
- 2 Yuuki Aoike, Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Kazuhiro Kurita, and Yota Otachi. An improved deterministic parameterized algorithm for cactus vertex deletion. *CoRR*, abs/2012.04910, 2020. [arXiv:2012.04910](https://arxiv.org/abs/2012.04910).
- 3 Benjamin Bergognoux, Édouard Bonnet, Nick Brettell, and O-Joung Kwon. Close relatives of feedback vertex set without single-exponential algorithms parameterized by treewidth. *Proc. IPEC 2020, LIPIcs*, 180(3):1–17, 2020.
- 4 Édouard Bonnet, Nick Brettell, O-Joung Kwon, and Dániel Marx. Parameterized vertex deletion problems for hereditary graph classes with a block property. *Proc. WG2016, LNCS*, 9941:233–244, 2016.
- 5 Andreas Brandstädt and Dieter Kratsch. On the restriction of some NP-complete graph problems to permutation graphs. *Proc. FCT 1985, LNCS*, 199:53–62, 1985.
- 6 Andreas Brandstädt and Raffaele Mosca. Maximum weight independent set for l -claw-free graphs in polynomial time. *Discrete Applied Mathematics*, 237:57–64, 2018.
- 7 Nina Chiarelli, Tatiana R. Hartinger, Matthew Johnson, Martin Milanič, and Daniël Paulusma. Minimum connected transversals in graphs: New hardness results and tractable cases using the price of connectivity. *Theoretical Computer Science*, 705:75–83, 2018.
- 8 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 9 Konrad K. Dabrowski, Carl Feghali, Matthew Johnson, Giacomo Paesani, Daniël Paulusma, and Paweł Rzażewski. On cycle transversals and their connected variants in the absence of a small linear forest. *Algorithmica*, 82(10):2841–2866, 2020.
- 10 Peter Gartland, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzażewski. Finding large induced sparse subgraphs in $C_{>t}$ -free graphs in quasipolynomial time. *Proc. STOC 2021, ACM*, pages 330–341, 2021.
- 11 Sudeshna Kolay, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Quick but odd growth of cacti. *Algorithmica*, 79:271–290, 2017.
- 12 Pranabendu Misra, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Parameterized algorithms for even cycle transversal. *Proc. WG 2012*, 7551:172–183, 2012.
- 13 Andrea Munaro. On line graphs of subcubic triangle-free graphs. *Discrete Mathematics*, 340(6):1210–1226, 2017.
- 14 Svatopluk Poljak. A note on stable sets and colorings of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 15:307–309, 1974.
- 15 O. Vornberger. Komplexität von Wegeproblemen in Graphen. *Reihe Theoretische Informatik*, 5, 1979.

Stabilization Bounds for Influence Propagation from a Random Initial State

Pál András Papp ✉

ETH Zürich, Switzerland

Roger Wattenhofer ✉

ETH Zürich, Switzerland

Abstract

We study the stabilization time of two common types of influence propagation. In majority processes, nodes in a graph want to switch to the most frequent state in their neighborhood, while in minority processes, nodes want to switch to the least frequent state in their neighborhood. We consider the sequential model of these processes, and assume that every node starts out from a uniform random state.

We first show that if nodes change their state for any small improvement in the process, then stabilization can last for up to $\Theta(n^2)$ steps in both cases. Furthermore, we also study the proportional switching case, when nodes only decide to change their state if they are in conflict with a $\frac{1+\lambda}{2}$ fraction of their neighbors, for some parameter $\lambda \in (0, 1)$. In this case, we show that if $\lambda < \frac{1}{3}$, then there is a construction where stabilization can indeed last for $\Omega(n^{1+c})$ steps for some constant $c > 0$. On the other hand, if $\lambda > \frac{1}{2}$, we prove that the stabilization time of the processes is upper-bounded by $O(n \cdot \log n)$.

2012 ACM Subject Classification Mathematics of computing → Graph coloring; Theory of computation → Distributed computing models; Theory of computation → Self-organization

Keywords and phrases Majority process, Minority process, Stabilization time, Random initialization, Asynchronous model

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.83

Related Version *Full Version:* <https://arxiv.org/abs/2107.02076>

1 Introduction

Dynamically changing colorings in a graph can be used to model various situations when entities of a network are in a specific state, and they occasionally decide to change their state based on the states of their neighbors. Such colorings are essentially a form of distributed automata, where the nodes can represent anything from brain cells to rival companies; as such, the study of these processes has applications in almost every branch of science.

One prominent example of such colorings is a *majority process*, where each node wants to switch to the color that is most frequent in its neighborhood. These processes are used to model a wide range of phenomena in social sciences, e.g. the spreading of political opinions in social networks, or the adoption of different social media platforms [16, 7, 20].

Another example is the dual setting of a *minority process*, where each node wants to switch to the least frequent color among its neighbors. Minority processes can model settings where nodes would prefer to differentiate from each other, e.g. frequency selection in wireless networks, or selecting a production strategy in a market economy [6, 21, 9].

In our paper, we analyze the stabilization time of majority and minority processes, i.e. the number of steps until no node wants to change its color anymore. We study the processes in the *sequential* (or asynchronous) model, where in every step, exactly one node switches its color. As such, stabilization time in the sequential model describes the total number of switches before the process terminates.



© Pál András Papp and Roger Wattenhofer;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 83; pp. 83:1–83:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Compared to a synchronous setting, the sequential model has the advantage that neighbors are never switching at the exact same time; this prevents the process from ending up in an infinitely repeating periodic pattern. This property is indeed a reasonable assumption in many application areas, including the examples mentioned above: you are highly unlikely to e.g. switch your wireless frequency at the exact same time as your neighbors, or change your political opinion at the exact same time as your friends.

We study the maximal stabilization time of the processes in general graphs, assuming that the initial coloring of nodes is chosen uniformly at random. This setting may be relevant for a worst-case analysis in applications where the only thing we can influence is the initial coloring. For example, a wireless service provider might have no control over the topology of the network or the times when clients decide to switch their frequency, but it could easily ensure that its devices are initialized with a randomly chosen frequency.

An important parameter of the model is the switching rule, i.e. the threshold at which a node decides to switch to the opposite color. Two very natural rules are (i) basic switching, when nodes decide to switch for any small improvement, and (ii) proportional switching, when we have a real parameter $\lambda \in (0, 1)$, and nodes only change their color if they are motivated to switch by a $\frac{1+\lambda}{2}$ fraction of their neighborhood.

In our paper, we study the stabilization time for both basic and proportional switching. As a warm-up (in Section 5), we first show that in case of basic switching, both minority and majority processes can take $\Omega(n^2)$ steps to stabilize with high probability, matching a naive upper bound of $O(n^2)$. This follows from an extension of the lower-bound construction in [27] to the random-initialized case.

Our main contributions (Sections 6 and 7) are stabilization bounds in case of proportional switching:

- for proportional switching with $\lambda < \frac{1}{3}$, we present a construction that w.h.p. exhibits a superlinear stabilization time of $\Omega(n^{1+c})$ for a constant $c > 0$ that depends on λ .
- for proportional switching with $\lambda > \frac{1}{2}$, we show that w.h.p. the process always stabilizes in $O(n \cdot \log n)$ steps, essentially matching a straightforward lower bound of $\Omega(n)$.

2 Related work

Majority and minority processes have been extensively studied from numerous different perspectives since the early 1980s [15, 11]. Most of the results focus on the simplest case of two colors, since this already captures the interesting properties of the process, and a generalization to more colors is often straightforward.

Many different variants of these processes have been inspired by application areas ranging from particle physics to social science, as in case of e.g. Ising systems or the voter model [23, 22]. In particular, there is extensive literature on more sophisticated process definitions that aim to provide a more realistic model for a specific application, such as social opinion dynamics or virus infection spreading [2, 1, 8, 25].

In case of majority processes, there is a particular interest in analyzing how a small set of nodes can influence the final state [35, 34, 14, 33, 3]. For both processes, there are also numerous works on the analysis of stable states [17, 5, 21, 18, 4]. However, in contrast to our work, most of these earlier results assume a synchronous setting, and only study the process on specific graph topologies, e.g. cliques, grids or Erdős-Rényi random graphs.

There is a recent line of work on stabilization time in general graphs; however, these results assume a worst-case initial coloring. For basic switching, the work of [12] shows that in the sequential adversarial and synchronous models, stabilization can last for $\tilde{\Omega}(n^2)$

steps, matching a straightforward upper bound of $O(n^2)$. A similar lower bound is known for minority processes [27]. On the other hand, the two processes exhibit very different behavior in a benevolent sequential model: majority processes always stabilize in $O(n)$ time, while minority processes can last for quadratically many steps [12, 27].

On the other hand, if we consider general graphs with proportional switching, then the sequential processes are known to exhibit a worst-case runtime between quadratic and linear, depending on the parameter λ of the switching rule [29]. Stabilization time in this case is characterized by a non-elementary function $f(\lambda)$ that monotonically and continuously decreases from 1 to 0 on the interval $[0, 1]$. The results of [29] show that for any $\varepsilon > 0$, stabilization time is upper-bounded by $O(n^{1+f(\lambda)+\varepsilon})$, and the process can indeed last for $\Omega(n^{1+f(\lambda)-\varepsilon})$ steps. Our results are an interesting contrast to this, showing that if we randomize the initial state, then the process can only take $\Omega(n^{1+c})$ steps for smaller λ values.

For general weighted graphs and a worst-case initial coloring, an exponential lower bound has also been shown for both majority [19] and minority [28] processes.

There are also various works that assume a randomized initial coloring, but these results focus on special classes of graphs. For majority processes, stabilization time from a randomized initial state has been analyzed in Erdős-Rényi random graphs, grids, tori and expanders [13, 26, 10, 24]. For minority processes, the works of [30, 31, 32] study stabilization in cliques, cycles, trees and tori. As such, to our knowledge, stabilization time from a randomized initial coloring has not yet been studied in general graphs.

3 Model definition and tools

3.1 Preliminaries

We study the processes on simple, unweighted, undirected graphs $G(V, E)$ with node set V and edge set E . We denote the nodes of the graph by u or v , and the number of nodes in the graph by n . For a specific node v , we denote the neighborhood of v by $N(v)$, and the degree of v by $d_v = |N(v)|$. For ease of presentation, we usually define the size of our graph constructions in terms of an (almost) linear parameter m , and in the end, we select a value of m that ensures $m \in \tilde{\Theta}(n)$.

As common in this area, we focus on the case of two colors. That is, we say that a *coloring* of the graph is a function $\gamma : V \rightarrow \{\text{black}, \text{white}\}$. For a specific coloring γ , we define $N_s(v) = \{u \in N(v) \mid \gamma(v) = \gamma(u)\}$ as the neighbors of v with the same color, and $N_o(v) = \{u \in N(v) \mid \gamma(v) \neq \gamma(u)\}$ as the neighbors of v with the opposite color.

We use the concept of *conflicts* to define both majority and minority processes in a general form. We say that there is a *conflict* on the edge (u, v) if this edge motivates v to change its color; more formally, if $u \in N_o(v)$ in case of a majority process, and if $u \in N_s(v)$ in case of a minority process. We use $N_c(v)$ to denote the conflicting neighbors of v under γ , i.e. $N_c(v) = N_o(v)$ for majority and $N_c(v) = N_s(v)$ for minority.

Given a specific coloring γ , we say that node v is *switchable* if $|N_c(v)|$ is larger than a specific threshold, which is defined by the so-called *switching rule* (discussed in detail in the next subsection). If v is switchable, then it can change its color to the opposite color (i.e. it can *switch*). We also use the word *balance* to refer to the metric $\frac{|N_c(v)|}{d_v}$ in general, which indicates how close node v is to being switchable.

A *majority/minority process* is a sequence of colorings of the graph G (known as *states*). Every state is obtained from the previous state by switching a switchable node in the previous state. We assume that exactly one node switches in each step, which is often known as the

sequential or asynchronous model of the process. In our paper, we also assume that the initial state of the process is a *uniform random coloring*, i.e. each node is white with probability $\frac{1}{2}$ and black with probability $\frac{1}{2}$, independently from other nodes.

We say that a state of the process is *stable* if there are no more switchable nodes in the graph. The number of steps in the process (from the initial state until a stable state is reached) is known as the *stabilization time* of the process.

We study the processes in general graphs, and we are interested in the longest possible stabilization time of a process, i.e. if in each step, the next node to switch among the switchable nodes is selected by an adversary who maximizes stabilization time. In other words, we study the worst-case stabilization of a graph on n nodes under the worst possible ordering of switches.

We also use basic tools from probability theory, such as the union bound and the Chernoff bound, and the concept of an event happening *with high probability* (*w.h.p.*). For completeness, a brief summary of these techniques is provided in the full version of the paper.

3.2 Switching rules

Another important parameter of the processes is the condition that allows nodes to switch their color. There are two natural candidates for such a switching rule:

► **I. Basic switching:** *node v is switchable if $|N_c(v)| > \frac{1}{2} \cdot d_v$.*

► **II. Proportional switching:** *node v is switchable if $|N_c(v)| \geq \frac{1+\lambda}{2} \cdot d_v$.*

Note that both rules ensure that the overall number of conflicts in the graph strictly decreases in each switching step. Since there are at most $|E| = O(n^2)$ conflicts in the graph initially, we obtain a straightforward upper bound of $O(n^2)$ on the stabilization time.

In case of basic switching, a node switches its color for an arbitrarily small improvement. Alternatively, if we denote the complement of $N_c(v)$ by $N_{\bar{c}}(v) := N(v) \setminus N_c(v)$, we can also formulate this rule as $|N_c(v)| - |N_{\bar{c}}(v)| > 0$. In case of the worst possible initial coloring, this rule is known to allow a stabilization time of $\Theta(n^2)$ [27, 12, 18].

In contrast to this, proportional switching is defined for a specific parameter $\lambda \in (0, 1]$, and it requires that v is in conflict with a specific portion of its neighborhood, with $\frac{1+\lambda}{2} \in (\frac{1}{2}, 1]$. This is often a more realistic approach if nodes have a large degree, or if switching also induces some cost in an application area. Equivalently, we can rephrase this rule as $|N_c(v)| - |N_{\bar{c}}(v)| \geq \lambda \cdot d_v$. This shows that whenever v switches, the total number of conflicts in the graph decreases by at least $\lambda \cdot d_v$, and v can have at most $\frac{1+\lambda}{2} \cdot d_v - \lambda \cdot d_v = \frac{1-\lambda}{2} \cdot d_v$ conflicts on the incident edges after the switch.

In case of a worst-case initial coloring, the maximal stabilization time for proportional switching is between quadratic and linear, following a monotonously decreasing non-elementary function $f(\lambda)$ described in [29]. Since this non-elementary function also plays a role in our lower bound, we discuss $f(\lambda)$ in the full version for completeness.

Note that for a very small λ value approaching 0, we can obtain basic switching as a special case of proportional switching in the limit.

3.3 Application of earlier results

We also apply the basic ideas behind some of the constructions from previous work, which were used to show similar lower bounds for a worst-case initial coloring.

Construction idea for basic switching. Recall that the result of [27] provides a quadratic lower bound on the stabilization time of minority processes.

► **Theorem** (from [27]). *Consider minority processes under the basic switching rule. There exists a class of graphs and an initial coloring with a stabilization time of $\Omega(n^2)$.*

The main idea of the construction is to have a set P of m nodes, attached to two further sets A and B of size m . The construction makes sure that every node in A and B wants to switch to the opposite color. Then we switch these nodes in an alternating fashion: one from A , one from B , one from A again, and so on. The set P is designed such that its neighborhood is approximately balanced, and thus after each of these steps, the entire set P is switchable. Switching P after each step gives a sequence of $m \cdot 2m = \Theta(n^2)$ switches.

Black box construction for proportional switching. We also use the result of [29], which provides a lower bound construction for any $\lambda \leq \frac{1}{3}$ in case of proportional switching and worst-case initial coloring. We apply this graph as a black box in our constructions, and refer to it as the PROP construction.

► **Theorem** (from [29]). *Consider majority/minority processes under proportional switching for any $\lambda \leq \frac{1}{3}$. There exists a class of graphs and an initial coloring with a stabilization time of $\Omega(n^{1+f(\lambda)-\epsilon})$ for the function f and for any $\epsilon > 0$.*

4 Basic observations

4.1 Initially balanced sets

Since we start from a uniform random initial coloring, a basic tool in our proofs is the fact that w.h.p., a large set of nodes has a balanced distribution of the colors initially.

► **Definition 1** (ϵ -balanced set). *Given a specific coloring, we say that a set of nodes S is ϵ -balanced if the number of white nodes in S is within $[(\frac{1}{2} - \epsilon) \cdot |S|, (\frac{1}{2} + \epsilon) \cdot |S|]$.*

► **Lemma 2.** *Let S_1, \dots, S_k be subsets of nodes in G such that $|S_i| \geq c_0 \cdot \log n$ for some constant c_0 for all $i \in \{1, \dots, k\}$, and $k \leq n$. Then for any constant $\epsilon > 0$, there is a c_0 such that w.h.p., each set S_i is initially ϵ -balanced.*

Proof. Let us select $c_0 = \frac{3}{2\epsilon}$. According to the Chernoff bound, the probability that S_i is not ϵ -balanced is at most

$$2 \cdot e^{-4\epsilon^2 \cdot \frac{1}{6} \cdot |S_i|} \leq 2 \cdot e^{-\frac{2}{3}\epsilon^2 \cdot c_0 \cdot \log n} = 2 \cdot n^{-2}.$$

If we take a union bound over all the $k \leq n$ subsets, the probability that any of them is not ϵ -balanced is at most $n \cdot 2 \cdot n^{-2} = 2 \cdot n^{-1}$, so w.h.p. the claim indeed holds. ◀

In particular, we can select a high constant c_0 , and refer to nodes v with $d_v \geq c_0 \cdot \log n$ as *high-degree* nodes, and the remaining nodes as *low-degree* nodes. Then Lemma 2 can be rephrased into the following claim:

► **Corollary 3.** *For any $\epsilon > 0$, there exists a c_0 such that w.h.p. the following claim holds: for all the high-degree nodes v in G , $N(v)$ is initially ϵ -balanced.*

4.2 Linear lower bound

Note that we can easily provide an example of linear stabilization time, even for proportional switching with any $\lambda \in (0, 1)$.

Consider an edge graph, i.e. a connected component with only two adjacent nodes u and v . With a probability of $\frac{1}{2}$, node v is initially switchable in this graph, for both majority/minority processes (since it has the opposite/same color as u , respectively). Let us take $\frac{n}{2}$ independent copies of this single-edge graph; this gives $\frac{n}{2}$ nodes in the role of v . Then $\frac{n}{4}$ of these nodes are switchable in expectation, and with a Chernoff bound, one can show that at least $\frac{n}{8}$ are switchable w.h.p.. We can switch these $\frac{n}{8}$ nodes in any order to obtain a sequence of $\frac{n}{8} \in \Omega(n)$ switches.

5 Lower bound constructions for basic switching

For basic switching, we can give an example of quadratic stabilization time by a suitable extension of the construction in [27] to the random-initialized setting.

In our analysis, we refer to a set of nodes as a *group* if they all have exactly the same neighborhood. In our figures, we denote groups by double-sided circles, with the cardinality shown beside the group, and an edge between two groups denotes a complete bipartite connection between the corresponding sets. Note that the nodes of a group always prefer the same color.

► **Theorem 4.** *Consider majority/minority processes under the basic switching rule, starting from a uniform random initial coloring. There exists a class of graphs that exhibit a stabilization time of $\Omega(n^2)$ with high probability in this model.*

We now outline the main ideas of these graphs, with the details discussed in the full version.

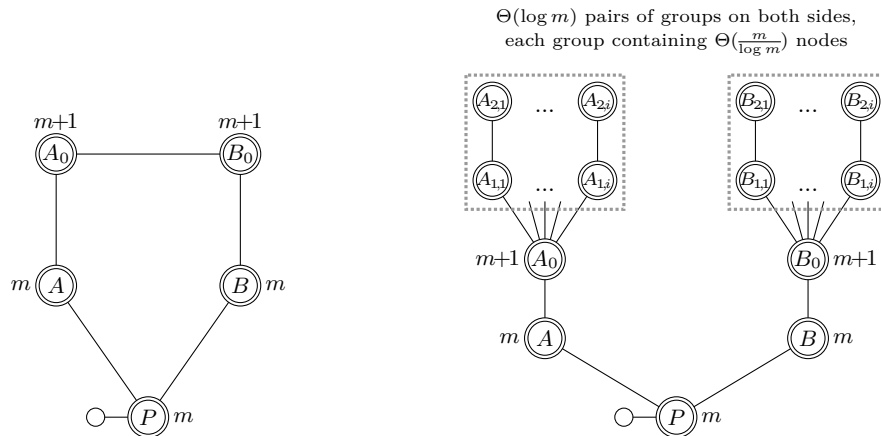
5.1 Minority processes

For minority processes, consider the graph in Figure 1, which is essentially an extension of the graph in [27] with a complete bipartite connection between A_0 and B_0 . For simplicity, we add an extra node to ensure that P has an odd degree. The graph has $5m + 3$ nodes, and thus $m \in \Theta(n)$.

Regardless of the initial coloring, each node in A_0 has the same preferred color, since they all have exactly the same neighbors and they have an odd degree. Thus we can switch each node in A_0 to this preferred color (if it did not have this color already). Assume w.l.o.g. that this color is white. Since now A_0 is white entirely, we can switch each node in B_0 to black. With this, the preferred color of each node in A becomes black, and the preferred color of each node in B becomes white.

An intuitive description of the remaining sequence is as follows. Both A and B have approximately $\frac{m}{2}$ nodes (and w.h.p. at least $\frac{m}{3}$ nodes) that have the same color as the group above. These nodes are now all switchable, regardless of the color of nodes in P . We disregard the remaining nodes, and only focus on these $\frac{m}{3}$ switchable nodes in A and B .

Initially, the neighborhood of P is w.h.p. ϵ -balanced. Hence by switching only $\epsilon \cdot m$ of nodes either in A or in B , we can ensure that P has exactly one more white neighbor than black, which allows us to switch the entire group P to black. Then by switching one node in A to black, P will have one more black neighbor than white, so P becomes switchable again. We can then switch the nodes in A and B in an alternating fashion; this ensures that P always has one more same-colored neighbor after each step, which makes P switchable again. This process allows us to switch the nodes of P altogether $\Theta(m)$ times, which already adds up to a sequence of $\Theta(m^2) = \Theta(n^2)$ switches.



■ **Figure 1** Lower bound constructions of $\Omega(n^2)$ steps in case of basic switching, for minority processes (left) and majority processes (right). Recall that double-sided circles denote groups, and edges between groups denote a complete bipartite connection between the two groups.

5.2 Majority processes

The case of majority processes is more involved, since in this case, it is more difficult to ensure that the groups A_0 and B_0 attain different colors.

Instead of connecting A_0 to B_0 , we connect A_0 to $\Theta(\log m)$ further groups of size $\Theta(\frac{m}{\log m})$, denoted by $A_{1,1}, A_{1,2}, \dots$. Finally, we add $\Theta(\log m)$ more distinct groups $A_{2,1}, A_{2,2}, \dots$, also on $\Theta(\frac{m}{\log m})$ nodes each, and we create a complete bipartite connection between $A_{1,i}$ and $A_{2,i}$. We attach the same structures to group B_0 in a symmetric manner; see Figure 1 for an overview of the construction.

The main idea of the construction is as follows. With probability $\frac{1}{2}$, the group $A_{1,i}$ has more white nodes than black initially, which allows us to switch $A_{2,i}$ entirely to white. Since the groups $A_{1,i}$ are independent, there is indeed w.h.p. an index \hat{i} such that the group $A_{2,\hat{i}}$ can be switched entirely to white. The neighbors of $A_{1,\hat{i}}$ are initially approximately balanced, so after recoloring all the $\Theta(\frac{m}{\log m})$ nodes in $A_{2,\hat{i}}$ to white, $A_{1,\hat{i}}$ has more white neighbors than black; this allows us to switch all of $A_{1,\hat{i}}$ to white. We note while our previous steps all follow directly from Corollary 3, this specific step requires a slightly stronger version of the Chernoff bound.

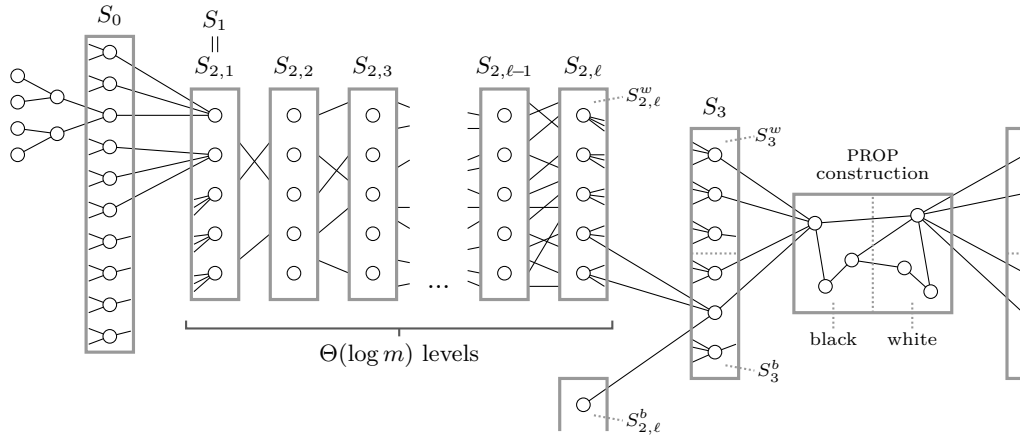
We can then apply a similar reasoning on the group A_0 : since it was w.h.p. balanced initially, and turning $A_{1,\hat{i}}$ to white has increased the number of its white neighbors by $\Theta(\frac{m}{\log m})$ w.h.p., we can also turn the entire group A_0 white. In a similar fashion, we can use groups $B_{2,\hat{i}}$ and $B_{1,\hat{i}}$ to switch each node in B_0 black w.h.p..

Once A_0 is white and B_0 is black, we again have $\Theta(m)$ switchable nodes in both A and B , and thus we can apply the same alternating method as in the minority case.

6 Proportional switching: lower bound for $\lambda < \frac{1}{3}$

We now show that for proportional switching with small λ values, stabilization time can indeed be superlinear. Note that $\lambda < \frac{1}{3}$ implies that $\frac{1+\lambda}{2} = \frac{2}{3} - \delta$ for some $\delta > 0$.

We present our lower bound construction for majority processes; however, since our graph is bipartite, we can easily adapt this result to minority processes by inverting the colors in one of the color classes. More details of this technique are available in the full version.



■ **Figure 2** High-level illustration of the proportional lower bound construction for any $\lambda < \frac{1}{3}$.

► **Theorem 5.** Consider majority/minority processes under the proportional switching rule for any $\lambda < \frac{1}{3}$, starting from a uniform random initial coloring. For any $\varepsilon > 0$, there exists a class of graphs that exhibit a stabilization time of $\Omega\left(n^{1+f\left(\frac{2-\lambda}{1-\lambda}\right)-\varepsilon}\right)$ with high probability.

In a simplified formulation, this means that there exists a constant $c > 0$ such that there is a construction with a stabilization time of $\Omega(n^{1+c})$ in this setting.

We divide our construction technique into five main *phases*, and discuss them separately. In each phase of the construction, we will refer to some edges of the nodes as *output* edges, which go to the following phase of the construction. In a specific phase, we always achieve a desired behavior without any change on these output neighbors yet. An overview of the entire construction is available in Figure 2.

As before, we define our construction in terms of a parameter $m = \tilde{\Theta}(n)$, and discuss the value of m in the end.

- First, in the *Opening Phase*, our goal is to create a set S_0 of constant-degree nodes such that (i) each node in S_0 has 1 output edge to the next phase, and (ii) for any parameter $p < 1$, we can switch each node in S_0 to black with a probability of at least p , independently from the remaining nodes.
- In the *Collection Phase*, we use our Opening Phase construction to produce another set S_1 where (i) each node in S_1 has $c_0 \cdot \log n$ output edges for a large enough constant c_0 , and (ii) w.h.p. we can switch all the nodes in S_1 to black.
- In the *Growing Phase*, we begin with this node set $S_{2,1} := S_1$, and add a range of further levels $S_{2,2}, S_{2,3}, \dots$ of the same size. Every level $S_{2,i}$ is only connected to the previous and next levels $S_{2,i-1}$ and $S_{2,i+1}$. The levels will have an exponentially increasing output degree, and hence in at most $\ell \approx \log m$ steps, we arrive at a final level $S_{2,\ell}$ where each node has an output degree of $\Theta(m)$. As in case of S_1 , we show that we can w.h.p. turn each node in $S_{2,i}$ (and finally, in $S_{2,\ell}$) black.
- In the *Control Phase*, we use $S_{2,\ell}$ to produce a set S_3 where each node still has an output degree of $\Theta(m)$. We will ensure that (i) there is a specific point in the process where each node in S_3 is switchable to black, and (ii) later, there is a specific point in the process where each node in S_3 is switchable to white.

■ Finally, in the *Simulation Phase*, we take an instance of the PROP construction, and we use our set S_3 to force each node in this construction to take the desired “initial” color. We can then simulate the behavior of PROP as a black box, which is known to provide a superlinear stabilization time from this artificially enforced worst-case initial coloring. In this section, we outline the main ideas behind each of these phase. More details of the construction are discussed in the full version.

We note that the second and third phases can be generalized to any λ up to $\frac{1}{2}$; however, there is no straightforward way to do this for the remaining phases.

6.1 Opening Phase

To construct the set S_0 , first consider a node v with $d_v = 3$: one neighbor labeled as an output, and two further neighbors u_1 and u_2 . Initially, we have an $\frac{1}{2}$ chance that v is already black. Even if v is not black initially, we can switch it black if both u_1 and u_2 are black initially: we have $\frac{1+\lambda}{2} < \frac{2}{3}$, so 2 black neighbors out of 3 are indeed enough to make v switchable. The probability that initially v is white but u_1 and u_2 are black is $(\frac{1}{2})^3 = \frac{1}{8}$, so altogether, we can turn v black with a probability of $p_1 = \frac{5}{8}$.

Now assume that we take two such nodes that can be switched black with probability $\frac{5}{8}$, we denote them by u'_1 and u'_2 , and we connect their outputs to a new node v' . Again, v' is already black initially with probability $\frac{1}{2}$; if not, we can turn v' black if both u'_1 and u'_2 are switched black, which happens with a probability of p_1^2 . This provides a black v' with a probability of $p_2 = \frac{1}{2} + \frac{1}{2} \cdot (\frac{5}{8})^2 = \frac{89}{128}$.

We can continue this in a recursive manner, always taking two copies of the previous construction, and connecting them to a new root node. After i steps, we end up with a full binary tree on $2^{i+1} - 1$ nodes. This provides a black root node with a probability of p_i , defined by the recurrence

$$p_0 = \frac{1}{2} \quad \text{and} \quad p_{i+1} = \frac{1}{2} + \frac{1}{2} \cdot p_i^2.$$

One can easily show that $\lim_{i \rightarrow \infty} p_i = 1$. Hence for any constant parameter $p < 1$, there is an i such that $p_i \geq p$, and thus creating i layers with this method ensures that we can switch the final node black with probability at least p .

In order to build our set S_0 , we can simply take $m_0 = |S_0|$ independent copies of this tree. Since p is a constant, i and the tree size $2^{i+1} - 1$ are also constants; thus the whole phase only requires $O(m_0)$ nodes altogether.

6.2 Collection Phase

Let us introduce a logarithmic parameter $d_0 = c_0 \cdot \log n$. Given our Opening Phase construction S_0 , our next step is to create a smaller set S_1 on $m_1 = \frac{1}{4 \cdot d_0} \cdot m_0$ nodes. Recall that all the m_0 nodes in S_0 had exactly 1 output edge; this allows us to connect each $v \in S_1$ to $4 \cdot d_0$ distinct nodes in S_0 . We also add d_0 further output edges to each $v \in S_1$ to provide a connection to the next phase.

Since each node in S_0 becomes black with probability p independently, a Chernoff bound shows that v has at least $(p - \epsilon) \cdot 4 \cdot d_0$ black neighbors in S_0 with a probability of $1 - O(n^{-2})$. This already makes v switchable to black, since $d_v = 5 \cdot d_0$, and thus for the appropriate p and ϵ values we have

$$\frac{(p - \epsilon) \cdot 4 \cdot d_0}{5 \cdot d_0} \approx \frac{4}{5} > \frac{2}{3} > \frac{1 + \lambda}{2}.$$

Applying a union bound over all nodes $v \in S_1$, we get that w.h.p. the entire set S_1 can be switched to black.

6.3 Growing Phase

Given our set S_1 from the Collection Phase, the next step is to iteratively build a range of levels $S_{2,i}$ for $i = 1, 2, \dots$. Each of these levels has the same size $|S_{2,i}| = m_1$, but on the other hand, their degrees increase exponentially: the output degree of each node in $S_{2,i+1}$ is always twice as big as the output degree of the nodes in $S_{2,i}$.

We achieve this by connecting every pair of subsequent levels as a regular bipartite graph. Let us begin with $S_{2,1} := S_1$. Recall that each node in S_1 has d_0 output edges, so $S_{2,1}$ and $S_{2,2}$ will form a d_0 -regular bipartite graph. We then connect $S_{2,2}$ and $S_{2,3}$ as a $2 \cdot d_0$ -regular bipartite graph, $S_{2,3}$ and $S_{2,4}$ as a $4 \cdot d_0$ -regular bipartite graph, and so on. Thus in any level, we have a value d such that each node has d edges to the previous and $2d$ edges to the next level, and this value d doubles with each new level. Since the degrees grow exponentially, after about $\log m_1$ levels, we reach a last level $S_{2,\ell}$ where the output degree is $\Theta(m_1)$.

We use an induction to prove that we can w.h.p. turn all nodes black in each $S_{2,i}$. This is already known for $S_{2,1} = S_1$ initially. In the general case, let v be an arbitrary node of $S_{2,i}$. Since each v has at least d_0 output edges to $S_{2,i+1}$, we can use Lemma 2 to show that the output neighborhood of every node is initially ϵ -balanced. This means that for any $v \in S_{2,i}$, at least $(\frac{1}{2} - \epsilon) \cdot 2d = (1 - 2\epsilon) \cdot d$ outputs are already black initially. Due to the induction, we can turn all the d remaining neighbors in $S_{2,i-1}$ black, altogether giving $(2 - 2\epsilon) \cdot d$ black neighbors of v . With $d_v = 3 \cdot d$, this amounts to a ratio of $\frac{2-2\epsilon}{3}$ black nodes in $N(v)$. Since we have $\frac{1+\lambda}{2} = \frac{2}{3} - \delta$, a sufficiently small choice of ϵ always ensures that this ratio is above $\frac{1+\lambda}{2}$, and thus v is switchable to black. Hence each node in $S_{2,i}$ can indeed be turned black, which completes our induction.

6.4 Control Phase

In the following Control Phase, we create a new set S_3 on m_3 nodes. The goal of this phase is to ensure that at a specific point in the process, each $v \in S_3$ switches to black, and then at a later point, each $v \in S_3$ is switchable to white.

In order to be able to initialize a PROP construction on m nodes in the final phase, each node in S_3 will have an output degree of m , for some parameter m . A detailed analysis shows that for a large constant $\alpha > 1$, a choice of $m_3 = \frac{1}{\alpha} \cdot m_1$ and $m = \frac{1}{2} \cdot m_3$ suffices for our purposes.

To achieve the desired switching behavior for S_3 , we first create two copies of the previous phases: one of them ending with a level $S_{2,\ell}^b$ on $\alpha \cdot m$ nodes where w.h.p. each nodes switches to black, and the other one ending with a last level $S_{2,\ell}^w$ on $2\alpha \cdot m$ nodes where w.h.p. each node switches to white in a symmetric manner. We connect each node in S_3 to every node in both $S_{2,\ell}^b$ and $S_{2,\ell}^w$. As a result, each $v \in S_3$ has a degree of $d_v = (3\alpha + 1) \cdot m$. Note that the output degree of both $S_{2,\ell}^b$ and $S_{2,\ell}^w$ is $\Theta(m_1) = \Theta(\alpha \cdot m_3)$, so for α large enough, they can indeed be connected to each node in S_3 .

Now consider the neighbors of a node $v \in S_3$. First $S_{2,\ell}^b$ becomes black and v 's neighborhood in $S_{2,\ell}^w$ is ϵ -balanced; this gives at least $\alpha \cdot m + (\frac{1}{2} - \epsilon) \cdot 2\alpha \cdot m = 2\alpha \cdot m \cdot (1 - \epsilon)$ black neighbors in $N(v)$, amounting to a $\frac{2\alpha \cdot (1-\epsilon)}{3\alpha+1}$ fraction of d_v . As $\frac{1+\lambda}{2} = \frac{2}{3} - \delta$, for a sufficiently small ϵ and sufficiently large α , we can ensure that this ratio is larger than $\frac{1+\lambda}{2}$, and thus v is indeed switchable. We switch each $v \in S_3$ to black at this point.

After this, we turn each node in $S_{2,\ell}^w$ white. Nodes in S_3 now have $2\alpha \cdot m$ white neighbors at least; this again ensures that each $v \in S_3$ is now switchable to white. However, for our purposes in the last phase, we will only switch half of the nodes in S_3 white at this point (denoted by S_3^w), and leave the remaining part black (denoted by S_3^b).

6.5 Simulation Phase

Finally, we use the PROP construction on m nodes to obtain superlinear stabilization time. Given a node v in PROP, assume w.l.o.g. that v is initially black in the example sequence of PROP; we can apply the same technique for white nodes in a symmetric manner.

Our main idea is to connect v to some new nodes in S_3^b and S_3^w . When S_3^b and S_3^w both switch to black, this allows us to switch v to its desired initial color (black). Then when S_3^w switches back to white, the new neighbors become balanced, and thus the switchability of v will again depend on its original neighbors within PROP. However, with these extra connections, the original $N(v)$ is now only a smaller fraction of v 's total neighborhood, so this only allows us to simulate PROP with a smaller parameter $\lambda' < \lambda$.

More specifically, if v has original degree d'_v within the PROP construction, then we connect v to $\frac{1}{2} \cdot \frac{1+\lambda}{1-\lambda} \cdot d'_v$ arbitrary nodes in both S_3^b and S_3^w . We point out that our choice of $m = \frac{1}{2} \cdot m_3$ is indeed sufficient for this: since $\lambda < \frac{1}{3}$ implies $\frac{1+\lambda}{1-\lambda} < 2$, every node in the PROP construction needs at most $\frac{1}{2} \cdot \frac{1+\lambda}{1-\lambda} \cdot d'_v < d'_v$ new edges to both S_3^b and S_3^w . Hence with $d'_v < m$ in the PROP construction, it is indeed enough to have m nodes in the sets S_3^b and S_3^w . Furthermore, since each node in S_3 has an output degree of m , we can also connect a node in S_3^b or S_3^w to as many nodes in the PROP construction as necessary.

With v connected to $\frac{1}{2} \cdot \frac{1+\lambda}{1-\lambda} \cdot d'_v$ nodes in both S_3^b and S_3^w , the new degree of v is now

$$d_v = \left(1 + \frac{1+\lambda}{1-\lambda}\right) \cdot d'_v = \frac{2}{1-\lambda} \cdot d'_v,$$

so v requires $\frac{1+\lambda}{2} \cdot d_v = \frac{1+\lambda}{1-\lambda} \cdot d'_v$ conflicts to be switchable. Hence when S_3^b and S_3^w are both switched black, this is already enough to switch v black, since the two sets provide $2 \cdot \frac{1}{2} \cdot \frac{1+\lambda}{1-\lambda} \cdot d'_v = \frac{1+\lambda}{1-\lambda} \cdot d'_v$ black neighbors to v together. Later S_3^w switches to white; then for the rest of the process, v has $\frac{1}{2} \cdot \frac{1+\lambda}{1-\lambda} \cdot d'_v$ neighbors of both colors in S_3 .

Let us now select $\lambda' = \frac{2\lambda}{1-\lambda}$, and apply the PROP construction for λ' as a black box. If v was switchable in the original PROP graph, it is also switchable in our construction. This allows us to run the entire sequence of $m^{1+f(\lambda')-\varepsilon}$ steps in PROP. Then together with the $\frac{1}{2} \cdot \frac{1+\lambda}{1-\lambda} \cdot d'_v$ additional conflicts to either S_3^b or S_3^w , v has at least $\frac{1+\lambda}{1-\lambda} \cdot d'_v = \frac{1+\lambda}{2} \cdot d_v$ conflicts in our construction, and thus it is indeed switchable.

Hence we can indeed simulate the behavior of PROP in our construction: whenever v is switchable in the original PROP graph, it is also switchable in our construction. This allows us to run the entire sequence of $m^{1+f(\lambda')-\varepsilon}$ steps in PROP, giving a sequence of $m^{1+f(\frac{2\lambda}{1-\lambda})-\varepsilon}$ steps in terms of our λ .

One can observe that our construction contains only $O(m \cdot \log m)$ nodes altogether, thus allowing a choice of $m = \Theta(\frac{n}{\log n})$. This results in about

$$n^{1+f(\frac{2\lambda}{1-\lambda})-\varepsilon} \cdot \log n^{-(1+f(\frac{2\lambda}{1-\lambda})-\varepsilon)}$$

steps for the PROP sequence in terms of n . Since such a PROP construction exists for any $\varepsilon > 0$, we can get rid of the second factor in this lower bound by simply applying the same proof with a smaller value $\hat{\varepsilon} < \varepsilon$. Thus the claim of Theorem 5 follows.

7 Proportional switching: upper bound for $\lambda > \frac{1}{2}$

We now show that with $\lambda = \frac{1}{2} + \delta$ for some $\delta > 0$, stabilization happens w.h.p. in $\tilde{O}(n)$ time. The only probabilistic element of this proof is the assumption that initially all high-degree nodes have an ε -balanced neighborhood; this indeed holds w.h.p., as we have seen before in Corollary 3.

83:12 Stabilization Bounds for Influence Propagation from a Random Initial State

The idea of the proof is that even though there might be $\Theta(n^2)$ conflicts in the graph initially, only a few of these conflicts can propagate through the graph. Let us call a conflict on edge (u, v) in our current coloring an *original conflict* if it has been on the edge since the beginning of the process, i.e. if every previous state (including the initial state) already had a conflict on (u, v) .

► **Definition 6** (Active/Rigid conflicts). *We say that a conflict on edge (u, v) is rigid if it is an original conflict, and both u and v are high-degree nodes. Otherwise, the conflict is active.*

Our proof is obtained as a result of three observations: that (i) there are only a few active conflicts in the graph initially, (ii) the number of active conflicts decreases in each step of the process, and (iii) the process stabilizes when there are no more active conflicts. Since the second point is the most complex out of the three claims, we first discuss it separately.

► **Lemma 7.** *The number of active conflicts strictly decreases in each step.*

Proof. Consider a specific step of the process, and let v be the node that switches in this step. Assume first that v is a low-degree node. In this case, v can only have active conflicts on its incident edges at any point in the process: initially, all conflicts of v are active by definition, and all the newly created conflicts in the process are also active. Since the number of conflicts on v 's incident edges decreases when v switches, the total number of active conflicts also decreases in this step.

Now assume that v is a high-degree node. Since $N(v)$ is initially ϵ -balanced, it has at most $(\frac{1}{2} + \epsilon) \cdot d_v$ rigid conflicts in the beginning, and since all the newly created conflicts in the process are active, it also has at most $(\frac{1}{2} + \epsilon) \cdot d_v$ rigid conflicts at any later point in the process. However, if v switches, then it must have at least $\frac{1+\lambda}{2} \cdot d_v$ incident conflicts; this implies that at least $\frac{1+\lambda}{2} \cdot d_v - (\frac{1}{2} + \epsilon) \cdot d_v$ of these conflicts are active. When v switches, it creates at most $\frac{1-\lambda}{2} \cdot d_v$ new (active) conflicts. Thus, to show that the number of active conflicts decreases, we only require

$$\frac{1+\lambda}{2} \cdot d_v - \left(\frac{1}{2} + \epsilon\right) \cdot d_v > \frac{1-\lambda}{2} \cdot d_v,$$

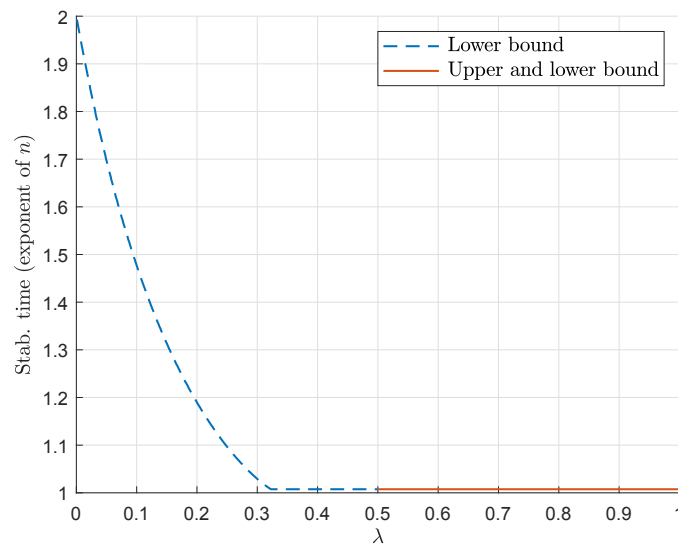
which is equivalent to $\lambda > \frac{1}{2} + \epsilon$. This holds for a sufficiently small choice of $\epsilon < \delta$. ◀

This already allows us to prove our upper bound.

► **Theorem 8.** *Consider majority/minority processes under the proportional switching rule for any $\lambda > \frac{1}{2}$, starting from a uniform random initial coloring. Any graph has a stabilization time of $O(n \cdot \log n)$ with high probability in this model.*

Proof. In any initial coloring, the number of active conflicts is at most $O(n \cdot \log n)$: each low-degree node has at most $c_0 \cdot \log n$ incident edges, and the number of low-degree nodes is at most n . Lemma 7 shows that the number of active conflicts decreases in each step, so there are no active conflicts in the graph after at most $O(n \cdot \log n)$ steps.

Once there are no more active conflicts, the coloring is stable, since nodes cannot be switchable without an active conflict on the incident edges. More specifically, due to the ϵ -balanced property, all high-degree nodes v have at most $(\frac{1}{2} + \epsilon) \cdot d_v$ rigid conflicts on the incident edges, which is smaller than $\frac{1+\lambda}{2} \cdot d_v$ if we have $\epsilon < \frac{\lambda}{2}$. Low-degree nodes, on the other hand, can never have rigid conflicts on the incident edges at all. Thus the process indeed stabilizes in $O(n \cdot \log n)$ steps. ◀



■ **Figure 3** Our upper and lower bounds on stabilization time in the proportional case.

8 Conclusion

Our results show that the behavior of the processes from a randomized initial coloring is rather straightforward in case of the basic switching rule: stabilization time can indeed tightly match the naive upper bound of $O(n^2)$.

However, in case of proportional switching, our work does leave some open questions. Figure 3 illustrates our upper and lower bounds for this case. The most apparent open question is the behavior of the process for the $\lambda \in [\frac{1}{3}, \frac{1}{2}]$ case; in this interval, we only have the straightforward lower bound of Section 4.2. While the figure gives the impression that stabilization time might also have a $\tilde{O}(n)$ upper bound in this case, it remains for future work to prove or disprove this claim.

Furthermore, even for $\lambda < \frac{1}{3}$ when stabilization is known to be superlinear, one might also be interested in devising upper bounds. Currently, the best known upper bound is that of $O(n^{1+f(\lambda)+\varepsilon})$ from [29], which even applies for the worst-case initial coloring.

References

- 1 Victor Amelkin, Francesco Bullo, and Ambuj K Singh. Polar opinion dynamics in social networks. *IEEE Transactions on Automatic Control*, 62(11):5650–5665, 2017.
- 2 Vincenzo Auletta, Ioannis Caragiannis, Diodato Ferraioli, Clemente Galdi, and Giuseppe Persiano. Generalized discrete preference games. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16*, page 53–59. AAAI Press, 2016.
- 3 Vincenzo Auletta, Diodato Ferraioli, and Gianluigi Greco. On the complexity of reasoning about opinion diffusion under majority dynamics. *Artificial Intelligence*, 284:103288, 2020.
- 4 Cristina Bazgan, Zsolt Tuza, and Daniel Vanderpooten. Complexity and approximation of satisfactory partition problems. In *International Computing and Combinatorics Conference*, pages 829–838. Springer, 2005.
- 5 Cristina Bazgan, Zsolt Tuza, and Daniel Vanderpooten. Satisfactory graph partition, variants, and generalizations. *European Journal of Operational Research*, 206(2):271–280, 2010.
- 6 Zhigang Cao and Xiaoguang Yang. The fashion game: Network extension of matching pennies. *Theoretical Computer Science*, 540:169–181, 2014.

83:14 Stabilization Bounds for Influence Propagation from a Random Initial State

- 7 Luca Cardelli and Attila Csikász-Nagy. The cell cycle switch computes approximate majority. *Scientific reports*, 2:656, 2012.
- 8 Carmen C Centeno, Mitre C Dourado, Lucia Draque Penso, Dieter Rautenbach, and Jayme L Szwarcftir. Irreversible conversion of graphs. *Theoretical Computer Science*, 412(29):3693–3700, 2011.
- 9 Jacques Demongeot, Julio Aracena, Florence Thuderoz, Thierry-Pascal Baum, and Olivier Cohen. Genetic regulation networks: circuits, regulons and attractors. *Comptes Rendus Biologies*, 326(2):171–188, 2003.
- 10 Michal Feldman, Nicole Immorlica, Brendan Lucier, and S. Matthew Weinberg. Reaching Consensus via Non-Bayesian Asynchronous Learning in Social Networks. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 192–208, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
- 11 Françoise Fogelman, Eric Goles, and Gérard Weisbuch. Transient length in sequential iteration of threshold functions. *Discrete Applied Mathematics*, 6(1):95–98, 1983.
- 12 Silvio Frischknecht, Barbara Keller, and Roger Wattenhofer. Convergence in (social) influence networks. In *International Symposium on Distributed Computing*, pages 433–446. Springer, 2013.
- 13 Bernd Gärtner and Ahad N Zehmakan. Color war: Cellular automata with majority-rule. In *International Conference on Language and Automata Theory and Applications*, pages 393–404. Springer, 2017.
- 14 Bernd Gärtner and Ahad N Zehmakan. Majority model on random regular graphs. In *Latin American Symposium on Theoretical Informatics*, pages 572–583. Springer, 2018.
- 15 Eric Goles and Jorge Olivos. Periodic behaviour of generalized threshold functions. *Discrete Mathematics*, 30(2):187–189, 1980.
- 16 Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.
- 17 Sandra M Hedetniemi, Stephen T Hedetniemi, KE Kennedy, and Alice A Mcrae. Self-stabilizing algorithms for unfriendly partitions into two disjoint dominating sets. *Parallel Processing Letters*, 23(01):1350001, 2013.
- 18 Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale. On the voting time of the deterministic majority process. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, 2016.
- 19 Barbara Keller, David Peleg, and Roger Wattenhofer. How even tiny influence can have a big impact! In *International Conference on Fun with Algorithms*, pages 252–263. Springer, 2014.
- 20 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.
- 21 Jeremy Kun, Brian Powers, and Lev Reyzin. Anti-coordination games and stable graph colorings. In *International Symposium on Algorithmic Game Theory*, pages 122–133. Springer, 2013.
- 22 Thomas M Liggett. *Stochastic interacting systems: contact, voter and exclusion processes*, volume 324. Springer Science & Business Media, 2013.
- 23 Barry M McCoy and Tai Tsun Wu. *The two-dimensional Ising model*. Courier Corporation, 2014.
- 24 Elchanan Mossel, Joe Neeman, and Omer Tamuz. Majority dynamics and aggregation of information in social networks. *Autonomous Agents and Multi-Agent Systems*, 28(3):408–429, 2014.
- 25 Arpan Mukhopadhyay, Ravi R Mazumdar, and Rahul Roy. Voter and majority dynamics with biased and stubborn agents. *Journal of Statistical Physics*, 181(4):1239–1265, 2020.

- 26 Ahad N Zehmakan. Opinion forming in Erdős-Rényi random graph and expanders. In *29th International Symposium on Algorithms and Computations*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, 2018.
- 27 Pál András Papp and Roger Wattenhofer. Stabilization Time in Minority Processes. In *30th International Symposium on Algorithms and Computation (ISAAC 2019)*, volume 149 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
- 28 Pál András Papp and Roger Wattenhofer. Stabilization Time in Weighted Minority Processes. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
- 29 Pál András Papp and Roger Wattenhofer. A General Stabilization Bound for Influence Propagation in Graphs. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 90:1–90:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
- 30 Damien Regnault, Nicolas Schabanel, and Éric Thierry. Progresses in the analysis of stochastic 2d cellular automata: A study of asynchronous 2d minority. In Luděk Kučera and Antonín Kučera, editors, *Mathematical Foundations of Computer Science 2007*, pages 320–332. Springer Berlin Heidelberg, 2007.
- 31 Damien Regnault, Nicolas Schabanel, and Éric Thierry. On the analysis of “simple” 2d stochastic cellular automata. In *International Conference on Language and Automata Theory and Applications*, pages 452–463. Springer, 2008.
- 32 Jean-Baptiste Rouquier, Damien Regnault, and Éric Thierry. Stochastic minority on graphs. *Theoretical Computer Science*, 412(30):3947–3963, 2011.
- 33 Grant Schoenebeck and Fang-Yi Yu. Consensus of interacting particle systems on Erdős-Rényi graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1945–1964. SIAM, 2018.
- 34 Ahad N Zehmakan. Target set in threshold models. *Acta Mathematica Universitatis Comenianae*, 88(3), 2019.
- 35 Ahad N Zehmakan. Tight bounds on the minimum size of a dynamic monopoly. In *International Conference on Language and Automata Theory and Applications*, pages 381–393. Springer, 2019.

Parameterized (Modular) Counting and Cayley Graph Expanders

Norbert Peyerimhoff ✉

Department of Mathematical Sciences, Durham University, UK

Marc Roth ✉ 

Merton College, University of Oxford, UK

Johannes Schmitt ✉ 

Mathematical Institute, University of Bonn, Germany

Jakob Stix ✉

Mathematical Institute, Goethe-Universität Frankfurt, Germany

Alina Vdovina ✉

School of Mathematics and Statistics, Newcastle University, UK

Abstract

We study the problem $\#EDGESUB(\Phi)$ of counting k -edge subgraphs satisfying a given graph property Φ in a large host graph G . Building upon the breakthrough result of Curticapean, Dell and Marx (STOC 17), we express the number of such subgraphs as a finite linear combination of graph homomorphism counts and derive the complexity of computing this number by studying its coefficients.

Our approach relies on novel constructions of low-degree Cayley graph expanders of p -groups, which might be of independent interest. The properties of those expanders allow us to analyse the coefficients in the aforementioned linear combinations over the field \mathbb{F}_p which gives us significantly more control over the cancellation behaviour of the coefficients. Our main result is an exhaustive and fine-grained complexity classification of $\#EDGESUB(\Phi)$ for minor-closed properties Φ , closing the missing gap in previous work by Roth, Schmitt and Wellnitz (ICALP 21).

Additionally, we observe that our methods also apply to modular counting. Among others, we obtain novel intractability results for the problems of counting k -forests and matroid bases modulo a prime p . Furthermore, from an algorithmic point of view, we construct algorithms for the problems of counting k -paths and k -cycles modulo 2 that outperform the best known algorithms for their non-modular counterparts.

In the course of our investigations we also provide an exhaustive parameterized complexity classification for the problem of counting graph homomorphisms modulo a prime p .

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms; Theory of computation \rightarrow Problems, reductions and completeness; Mathematics of computing \rightarrow Combinatorics; Mathematics of computing \rightarrow Graph theory

Keywords and phrases Cayley graphs, counting complexity, expander graphs, fine-grained complexity, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.84

Related Version *Full Version:* <https://arxiv.org/abs/2104.14596>

Funding *Johannes Schmitt:* The third author was supported by the SNF Early Postdoc.Mobility grant 184245 and thanks the Max Planck Institute for Mathematics in Bonn for its hospitality.

Acknowledgements The second author is grateful to Holger Dell for fruitful discussions on the connections between [10] and our work.



© Norbert Peyerimhoff, Marc Roth, Johannes Schmitt, Jakob Stix, and Alina Vdovina; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 84; pp. 84:1–84:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Extended Abstract

In this work we study the problem of counting small patterns in large host graphs. With applications in a diverse set of disciplines such as constraint satisfaction problems [13, 7], database theory [15, 8] and network science [34, 1, 41], it is unsurprising that this problem has received significant attention from the viewpoint of parameterized and fine-grained complexity theory in recent years [2, 17, 33, 9, 12, 11, 5, 37, 38, 28, 39].

We continue this line of work and study the problem of counting k -edge subgraphs that satisfy a graph property Φ : For any fixed Φ , the problem $\#\text{EDGESUB}(\Phi)$ asks, on input a graph G and a positive integer k , to compute the number of (not necessarily induced) subgraphs with k edges in G that satisfy Φ . In particular, we focus on instances in which k is significantly smaller than G . Formally, we choose k to be the *parameter* of the problem and ask for which Φ there is a function f such that $\#\text{EDGESUB}(\Phi)$ can be solved in time $f(k) \cdot |V(G)|^{O(1)}$; in this case we call the problem *fixed-parameter tractable* with respect to the parameter k .

If $\#\text{EDGESUB}(\Phi)$ is not fixed-parameter tractable, it is desirable to improve the exponent of $|V(G)|$ in the running time as far as possible. For example, the best known algorithm for counting k -edge subgraphs [11] can be used to solve $\#\text{EDGESUB}(\Phi)$ in time $f(k) \cdot |V(G)|^{0.174k+o(k)}$ [39]. Additionally, it was shown in recent work that $\#\text{EDGESUB}(\Phi)$ is fixed-parameter tractable whenever Φ has *bounded matching number*, that is, whenever there is a constant upper bound on the size of the largest matching of any graph satisfying Φ [39]. If, for each k , the property Φ is true for only one graph on k edges, then the previous fixed-parameter tractability result is best possible: In this case, $\#\text{EDGESUB}(\Phi)$ becomes an instance of the counting version of the parameterized subgraph isomorphism problem which has been fully classified by Curticapean and Marx [12].

However, for arbitrary Φ , much less is known about the complexity of $\#\text{EDGESUB}(\Phi)$. In [39], two of the authors, together with Wellnitz, presented first results for more general properties such as connectivity, Eulerianity and, in particular, an *almost* exhaustive classification for minor-closed properties Φ , leaving (partially) open the case of forbidden minors of degree at most 2. In this work, we close this gap and provide a full dichotomy result:

► **Theorem 1.** *Let Φ be a minor-closed graph property. If Φ is trivially true or of bounded matching number, then $\#\text{EDGESUB}(\Phi)$ is fixed-parameter tractable. Otherwise, $\#\text{EDGESUB}(\Phi)$ is $\#\text{W}[1]$ -hard and, assuming the Exponential Time Hypothesis, it cannot be solved in time*

$$f(k) \cdot |G|^{o(k/\log k)}$$

for any function f .

Here, $\#\text{W}[1]$ is the parameterized counting analogue of NP; a formal definition is provided in the full version. Particular cases for which we obtain novel intractability results are given by the following (minor-closed) properties; the formal intractability results are stated and proved in the full version as well.

- $\Phi(H) = 1$ if H is a forest.
- $\Phi(H) = 1$ if H is a linear forest.
- $\Phi(H) = 1$ if the tree-depth of H is bounded by a constant.
- $\Phi(H) = 1$ if the Colin de Verdière Invariant of H is bounded by a constant.

Additionally, we investigate the property of being bipartite. For this case, we present not only a novel fine-grained lower bound, but also a $\#\text{W}[1]$ -hardness result, which was not known before.

► **Theorem 2.** *Let Φ be the property of being bipartite. Then $\#\text{EDGESUB}(\Phi)$ is $\#\text{W}[1]$ -hard and, assuming the Exponential Time Hypothesis, it cannot be solved in time*

$$f(k) \cdot |G|^{o(k/\log k)}$$

for any function f .

Our hardness results crucially rely on a novel construction of families of low-degree Cayley graph expanders of p -groups, which might be of independent interest. We will present the new Cayley graph expanders in the following theorem; their construction, as well as their role in the hardness proofs for $\#\text{EDGESUB}(\Phi)$ will be elaborated on in the technical discussion.

► **Theorem 3.** *Let $p \geq 3$ be a prime number, and $d \geq 2$ be an integer. We assume that $d \geq (p+3)/2$ if $p \geq 7$.*

Then there is an explicit construction of a sequence of finite p -groups Γ_i of orders that tend to infinity, with symmetric generating sets S_i of cardinality $2d$ such that the Cayley graphs $\mathcal{C}(\Gamma_i, S_i)$ form a family of expanders (of fixed valency $2d$ on a set of vertices of p -power orders and with vertex transitive automorphism groups).

Our methods do not only apply to exact counting, but also to modular pattern counting problems: Here the goal is to compute the number of occurrences of the pattern *modulo a fixed prime p* . In classical complexity theory, the study of modular counting problems has a rich history, such as the algorithm for computing the permanent modulo 2^ℓ [44], the so-called accidental algorithms [45], Toda's Theorem [43], classifications for modular $\#\text{CSPs}$ and Holants [24, 25] and the line of research on the modular homomorphism counting problem [16, 20, 21, 22, 26, 18, 27], only to name a few.

While results are scarcer, the *parameterized* complexity of modular (pattern) counting problems has also been studied in recent years [4, 14, 10], and we contribute to this line of research as follows: First, we provide a novel intractability result for modular counting of forests and matroid bases. We write $\#_p\text{FORESTS}$ for the problem of, given a graph G and a positive integer k , computing the number of forests with k edges in G , modulo p . Similarly, we write $\#_p\text{BASES}$ for the problem of, given a linear matroid M of rank k in matrix representation, computing the number of bases of M , modulo p ; the parameter of both problems is given by k .

► **Theorem 4.** *For each prime $p \geq 3$, the problems $\#_p\text{FORESTS}$ and $\#_p\text{BASES}$ are $\text{Mod}_p\text{W}[1]$ -hard and, assuming the randomised Exponential Time Hypothesis, cannot be solved in time $f(k) \cdot |G|^{o(k/\log k)}$ (resp. $f(k) \cdot |M|^{o(k/\log k)}$), for any function f .*

Here, $\text{Mod}_p\text{W}[1]$ is the parameterized modular counting version of NP. Roughly speaking, a problem is $\text{Mod}_p\text{W}[1]$ if it is at least as hard as counting k -cliques modulo p ; we give a formal definition in the full version. Additionally, we provide an algorithmic result for counting k -paths and k -cycles modulo 2:

► **Theorem 5.** *The problems of counting k -paths and k -cycles in a graph G modulo 2 can be solved in time $k^{O(k)} \cdot |V(G)|^{k/6+O(1)}$.*

We emphasize that the algorithm in the previous theorem is faster than the best known algorithms for (non-modular) counting of k -cycles/ k -paths, which run in time $k^{O(k)} \cdot |V(G)|^{13k/75+o(k)}$ [11]. Furthermore, it follows from a result by Curticapean, Dell and Husfeldt [10] that counting k -paths modulo 2 is $\text{Mod}_2\text{W}[1]$ -hard, implying that we cannot hope for an algorithm for k -paths running in time $f(k) \cdot |V(G)|^{O(1)}$. Finally, we study the parameterized complexity of counting homomorphisms modulo p . In the classical setting, the related problem of modular counting homomorphisms with *right-hand side* restrictions received much attention: For any fixed graph H , the problem $\#_p\text{HOMSTO}(H)^1$ asks, on input a graph G , to compute the number of homomorphisms from G to H modulo p . Despite significant effort [16, 20, 21, 22, 26, 18, 27], the problem has not been fully classified for each graph H .²

In this work, we consider the related *left-hand side* version of the problem. Adapting the definitions of Grohe, Dalmau and Jonsson [23, 13] for detecting and exact counting of homomorphisms, we define a problem $\#_p\text{HOM}(\mathcal{H})$ for each class of graphs \mathcal{H} and for each prime p : This problem expects as input a graph $H \in \mathcal{H}$ and an arbitrary graph G , and the goal is to compute the number of homomorphisms from H to G , modulo p . The problem is parameterized by the size of H , that is, we assume H to be significantly smaller than G .

It is known that the decision version $\text{HOM}(\mathcal{H})$ is fixed-parameter tractable (even solvable in polynomial time) if the treewidth of the cores of \mathcal{H} is bounded by a constant, and $\text{W}[1]$ -hard otherwise [23]. Similarly, the (exact) counting version $\#\text{HOM}(\mathcal{H})$ is known to be fixed-parameter tractable (even solvable in polynomial time) if the treewidth of the graphs of \mathcal{H} is bounded by a constant, and $\#\text{W}[1]$ -hard otherwise [13].

In case of counting modulo p , we establish an exhaustive classification for $\#_p\text{HOM}(\mathcal{H})$ along what we call the *p -reduced quotients* of the graphs in \mathcal{H} . Let H be a graph and let α be an automorphism of H of order p . Then we define the quotient graph H/α to have a vertex for each orbit of the action of α on $V(H)$, and two vertices corresponding to orbits \mathcal{O}_1 and \mathcal{O}_2 are made adjacent if and only if there are vertices $v_1 \in \mathcal{O}_1$ and $v_2 \in \mathcal{O}_2$ such that $\{v_1, v_2\} \in E(H)$. This induces a finite (possibly trivial) sequence $H = H_1, \dots, H_\ell$ where for $i = 1, \dots, \ell - 1$ we set $H_{i+1} = H_i/\alpha_i$ for some automorphism α_i of order p of H_i and where the last graph H_ℓ does not have an automorphism of order p . Then $H_p^* := H_\ell$ is called the *p -reduced quotient* of H .³ We will see that H_p^* is well-defined by proving that each of the aforementioned sequences yields the same graph, up to isomorphism.

Let us now state our classification for $\#_p\text{HOM}(\mathcal{H})$. In what follows, given a class \mathcal{H} , we write \mathcal{H}_p^* for the p -reduced quotients without self-loops of graphs in \mathcal{H} . We first present the algorithmic part:

► **Theorem 6.** *Let $p \geq 2$ be a prime and let \mathcal{H} be a class of graphs. The problem $\#_p\text{HOM}(\mathcal{H})$ can be solved in time*

$$\exp(\text{poly}(|V(H)|)) \cdot |V(G)|^{\text{tw}(H_p^*)+O(1)}.$$

In particular, $\#_p\text{HOM}(\mathcal{H})$ is fixed-parameter tractable if the treewidth of \mathcal{H}_p^ is bounded.*

Here tw denotes treewidth.

¹ For $p = 2$, the problem is usually denoted by $\oplus\text{HOMSTO}(H)$.

² However, very recently a full classification was announced by Bulatov and Kazemina [6].

³ We remark that H_p^* is related to the notion of involution-free reductions used in the analysis of the right-hand side version of the problem [16, 22]. However, the difference is that the p -reduced quotient identifies non-fixed points of an order- p automorphism by including a vertex for each orbit, while the involution-free reduction just deletes all non-fixed points.

► **Remark 7.** Using the quasi-polynomial time algorithm for GI due to Babai [3], we will also show how the algorithm in the previous theorem can be improved to run in quasi-polynomial time in the input length $|V(H)| + |V(G)|$. Additionally, proving that the construction of the p -reduced quotient is at least as hard as the graph automorphism problem, we observe that a polynomial-time algorithm is unlikely, unless the construction of the p -reduced quotients can be avoided.

For the intractability part of our classification, we show that unbounded treewidth of \mathcal{H}_p^* yields hardness:

► **Theorem 8.** *Let $p \geq 2$ be a prime and let \mathcal{H} be a computable class of graphs. If the treewidth of \mathcal{H}_p^* is unbounded, then $\#_p\text{HOM}(\mathcal{H})$ is $\text{Mod}_p\mathbb{W}[1]$ -hard and, assuming the randomised Exponential Time Hypothesis, cannot be solved in time*

$$f(|H|) \cdot |G|^{o(\text{tw}(H_p^*)/\log \text{tw}(H_p^*))}$$

for any function f .

Can You Beat Treewidth?

We conclude the presentation of our results by commenting on the factor of $1/(\log \dots)$ in the exponents of all of our fine-grained lower bounds. This factor is related to the conjecture of whether it is possible to “beat treewidth” [31]. In particular, we point out that the factor can be dropped in *all* of our lower bounds if this conjecture, formally stated as Conjecture 1.3 in [32], is true.

Technical Overview

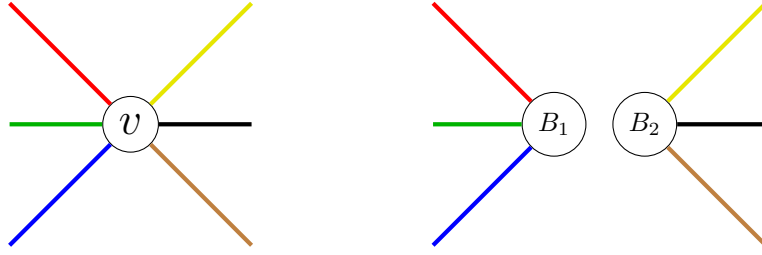
Our central approach follows the so-called Complexity Monotonicity framework due to Curticapean, Dell and Marx [11]. We express the counting problems considered in this work as formal linear combination of homomorphism counts, which allows us to derive the complexity of the problem at hand by analyzing the coefficients.

More precisely, let us fix a graph property Φ and a positive integer k . Given a graph G , we furthermore write $\#\text{EdgeSub}(\Phi, k \rightarrow G)$ for the number of k -edge subgraphs of G that satisfy Φ . It was shown in [39] that there exists a function of finite support $a_{\Phi, k}$ from graphs to rationals such that for every graph G we have

$$\#\text{EdgeSub}(\Phi, k \rightarrow G) = \sum_H a_{\Phi, k}(H) \cdot \#\text{Hom}(H \rightarrow G), \quad (1)$$

where $\#\text{Hom}(H \rightarrow G)$ is the number of graph homomorphisms from H to G . Curticapean, Dell and Marx [11] have shown that computing a linear combination as in (1) is *precisely as hard as* computing its hardest term. Fortunately, the complexity of counting and detecting homomorphisms from H to G is thoroughly classified [13, 31]: Roughly speaking, the higher the treewidth of H , the harder it is to compute the number of homomorphisms from H to G . Therefore, proving hardness of computing $\#\text{EdgeSub}(\Phi, k \rightarrow G)$ reduces to the purely combinatorial problem of determining which of the coefficients $a_{\Phi, k}(H)$ in (1) for high-treewidth graphs H are non-zero.

Unfortunately, it has turned out that the coefficients of such linear combinations for related pattern counting problems are often determined by (or even equal to) a variety of algebraic and topological invariants, whose analysis is known to be a difficult problem in its own right. For example, in case of the vertex-induced subgraph counting problem, the



■ **Figure 1** Illustration of the construction of a fractured graph. The left picture shows a vertex v of a graph H with incident edges $E_H(v) = \{\bullet, \bullet, \bullet, \bullet, \bullet\}$. The right picture shows the splitting of v in the construction of the fractured graph $H\#\sigma$ for a fracture σ satisfying that the partition σ_v contains two blocks $B_1 = \{\bullet, \bullet, \bullet\}$, and $B_2 = \{\bullet, \bullet, \bullet\}$.

coefficient of the clique is the reduced Euler characteristic of a simplicial graph complex [37], the coefficient of the biclique is the so-called alternating enumerator [14], and, more generally, the coefficients of dense graphs are related to the h - and f -vectors associated with the property of the patterns that are to be counted [38]. In all of the previous works mentioned here, the complexity analysis of the respective pattern counting problems therefore amounted to understanding the cancellation behaviour of those invariants. To do so, the papers used tools from combinatorial commutative algebra and, to some extent, topological fixed-point theorems.

In case of $\#\text{EDGESUB}(\Phi)$, two of the authors, together with Wellnitz, observed that the coefficients of high-treewidth low-degree vertex-transitive graphs can be analysed much easier than generic graphs of high treewidth such as the clique or the biclique [39]. First, it was shown that the coefficient of a graph H with k edges in (1) is equal to the *indicator* of Φ and H , defined as follows:⁴

$$a(\Phi, H) := \sum_{\sigma \in \mathcal{L}(\Phi, H)} \prod_{v \in V(H)} (-1)^{|\sigma_v| - 1} (|\sigma_v| - 1)!. \quad (2)$$

Here, $\mathcal{L}(\Phi, H)$ is the set of *fractures* σ of H such that the associated *fractured graph* $H\#\sigma$ satisfies Φ . Here, a fracture of a graph H is a tuple $\sigma = (\sigma_v)_{v \in V(H)}$, where σ_v is a partition of the set of edges $E_H(v)$ of H incident to v . Given a fracture ρ of H , the fractured graph $H\#\rho$ is obtained from H by splitting each vertex $v \in V(H)$ according to σ_v ; an illustration is provided in Figure 1.

As a consequence, the $\#\text{W}[1]$ -hardness results of Theorems 1 and 2 can be obtained if we find a family of graphs H of unbounded treewidth, such that $a(\Phi, H) \neq 0$ for infinitely many graphs H in this family. The almost tight conditional lower bound under the Exponential Time Hypothesis will, additionally, require sparsity of the graphs. In combination with the main observation in [39], stating that the indicator $a(\Phi, H)$ can be analysed much easier for vertex-transitive graphs, we propose that regular Cayley graph expanders are the right choice for the family of graphs to be considered. Indeed, those graphs are sparse, have high treewidth and are always vertex transitive. A particular family of Cayley graph expanders was already used in [39], but it turned out to be impossible to prove Theorems 1 and 2 relying only on this family of Cayley graph expanders; we discuss this in detail in the full version.

⁴ To be precise, the identity in (2) was obtained for a coloured version of $\#\text{EDGESUB}(\Phi)$. However, we will mostly rely on this result in a blackbox manner; all details of the coloured version necessary for the treatment in this paper will be carefully introduced when needed.

In this work, we therefore present novel constructions of families of low-degree Cayley graph expanders. Those will not only allow us to prove most of our main theorems by analysing their indicators, but might be of independent interest. For the sake of presentation, we decided to encapsulate the treatment of our constructions in separate sections, both in the extended abstract and the main part of the paper. We hope that this makes the paper accessible both for readers primarily interested in the novel construction of Cayley graph expanders, as well as for readers mainly interested in the analysis of the pattern counting problems. In particular, this last group may safely skip the next subsection and rely only on Theorem 3.

Construction of Low-Degree Cayley Graph Expanders

We prove Theorem 3 via an explicit construction of the groups Γ_i and the symmetric generating sets S_i in the full version, motivated by number theoretic objects. In what follows we present an overview of our construction.

Let us fix a prime $p \geq 3$. The starting point is an explicit arithmetic lattice (a discrete subgroup) in a group of generalized quaternions over a function field in characteristic p . The quaternion algebra is at the heart of the mathematical properties of extracting the finite p -groups and the expansion property of the resulting Cayley graphs, but it is not crucial for understanding the construction. Concretely, for any choice of elements $\alpha \neq \beta \in \mathbb{Z}/(p-1)\mathbb{Z}$ we construct an infinite group $\Gamma_{p;\alpha,\beta}$ defined in terms of $2(p+1)$ generators a_k, b_j (where the indices k, j run through sets $K, J \subseteq \mathbb{Z}/(p^2-1)\mathbb{Z}$ defined depending on α, β) and relations of length 4. The set of relations is described by explicit algebraic equations in the field \mathbb{F}_{p^2} . In [40] these groups were realized by mapping the generators a_k, b_j to explicit generalized quaternions, leading ultimately to an explicit injective group homomorphism

$$\Psi: \Gamma_{p;\alpha,\beta} \rightarrow \mathrm{GL}_3(\mathbb{F}_p[[t]]). \tag{3}$$

In other words, every element of $\Gamma_{p;\alpha,\beta}$ is sent to an invertible 3×3 -matrix whose entries are power series in some formal variable t , whose coefficients live in the finite field \mathbb{F}_p with p elements. This is made explicit for $p = 3$ in the full version, but could also be made explicit for any $p \geq 5$. Since the applications do not depend on concrete matrices, we merely state its existence.

To construct the finite p -groups Γ_i , consider the group homomorphism

$$\pi_i: \mathrm{GL}_3(\mathbb{F}_p[[t]]) \rightarrow \mathrm{GL}_3(\mathbb{F}_p[t]/(t^{i+1}))$$

taking a matrix with power series entries and truncating the power series after the term of order t^i . Then the group $\mathrm{GL}_3(\mathbb{F}_p[t]/(t^{i+1}))$ is finite, and we define Γ_i to be the image of the group $\Gamma_{p;\alpha,\beta}$ under the composition $\pi_i \circ \Psi$. These groups Γ_i are easily shown to be p -groups and they are what is called congruence quotients (by construction). The generators a_k, b_j from the construction of $\Gamma_{p;\alpha,\beta}$ map to symmetric generating sets T_i of Γ_i , i.e., to the set of cosets $a_k N_i, b_j N_i$ when $\Gamma_i = \Gamma_{p;\alpha,\beta}/N_i$ is considered as a factor group. Using results from [40], we know that the Cayley graphs $G_i = \mathcal{C}(\Gamma_i, T_i)$ associated to the congruence quotient groups Γ_i with respect to the generating sets T_i are expanders. This argument is worked out in [40] by Rungtanapirom and two of the authors, and it is based on a similar approach in the classical papers by Lubotzky, Phillips and Sarnak [30] and by Morgenstern [35]. We note here that the results of [40] ultimately rely on deep number theoretic results, namely a translation of the spectrum of the adjacency operator into Satake parameters of an associated automorphic representation and most crucially on work of Drinfeld on the geometric Langlands programme for GL_2 .

At this point we have proven Theorem 3 for the particular valency $2d = 2(p + 1)$. In order to obtain the more general valencies stated in the theorem (which will be crucial for some of our reductions), we recall in Section 3.1 of the full version that a uniformly controlled change of the generating sets T_i of the groups Γ_i (the generators must be mutually expressible in words of uniformly bounded length) preserves the expander property. This change of generating set is best performed by finding a smaller generating set for the underlying infinite group $\Gamma_{p;\alpha,\beta}$. This is done in Proposition 3.7 in the full version, reducing to $d = (p + 3)/2$ for all $p \geq 3$. The reduction is based on the explicit form of the relations and a combinatorial group theoretic result from [42] on the local permutation structure of the underlying geometric square complex. To improve even further for $p = 3$ we consider in Section 3.3 of the full version a concrete presentation of $\Gamma_{3;0,1}$ which is shown to reduce to 2 generators. For $p = 5$, an explicit example given in the full version achieves a reduction to 2 generators for $\Gamma_{5;0,2}$. In the last step, we will then show that by adding generators (as necessary) we obtain Theorem 3 for all d 's in the range that the theorem promises.

While this is not needed for the purposes of our hardness results, all of the constructions above are explicit, certainly in the weak sense that for a fixed p , the sequence of graphs G_i from Theorem 3 is computable. We also would like to emphasize again, that the expanders constructed for the proof of Theorem 3 consist of vertex transitive graphs, of prime power number of vertices, with a fairly low bound on the degree. All of this is made possible by working with very specific generalized quaternion groups in positive characteristic.

Analysis of the Indicators

Having established the existence of the low-degree Cayley graph expanders, we turn back to the analysis of the indicator

$$a(\Phi, H) = \sum_{\sigma \in \mathcal{L}(\Phi, H)} \prod_{v \in V(H)} (-1)^{|\sigma_v|-1} (|\sigma_v| - 1)! \tag{4}$$

Recall that we claimed the analysis of $a(\Phi, H)$ to be easier for vertex-transitive graphs. Let us now elaborate on this claim. First of all, we restate the formal definition of Cayley graphs for readers who skipped the explicit construction of our expanders: the *Cayley graph* of a group Γ together with a symmetric generating set⁵ $S \subseteq \Gamma$ is the graph $G = \mathcal{C}(\Gamma, S)$ with vertex set $V(G) = \Gamma$ and edge set

$$E(G) = \{(x, xs) \in V(G) \times V(G); x \in \Gamma, s \in S\}.$$

Since S is symmetric, with any edge (x, xs) the Cayley graph also contains the edge with opposite orientation $(xs, x) = (xs, (xs)s^{-1})$. Hence we consider Cayley graphs as the underlying unoriented graph.

Given a Cayley graph G as above, the group Γ acts on the graph by letting $g \in \Gamma$ send the vertex $v \in V(G) = \Gamma$ to gv . This action extends to the set of fractures $\mathcal{L}(\Phi, H)$ and since the terms $\prod_{v \in V(H)} (-1)^{|\sigma_v|-1} (|\sigma_v| - 1)!$ in the formula (4) are shown to be invariant under this action, the group Γ naturally permutes these summands. Since our Cayley graph expanders G_i arise from p -groups Γ_i , it follows that when evaluating the indicator $a(\Phi, G_i)$ modulo p , only those contributions from fractures fixed under Γ_i survive. Now recall that σ_v is a partition of the edges incident to v . The fixed-point fractures σ will satisfy that all σ_v are equal if we identify the edges incident to v with the elements of the generating set. Since, for

⁵ This means a subset $S \subseteq \Gamma$ of the group that generates this group and satisfies $S^{-1} = S$.

fixed p , our Cayley graph expanders have constant degree, we can thus prove the indicator to be non-zero modulo p by considering just a constant number of fractures. This approach was first used in [39], and we will show that it becomes significantly more powerful if applied to our novel Cayley graph expanders.

Now let us illustrate and sketch this approach for the property Φ of being bipartite, that is, for proving Theorem 2. By Theorem 3, there is a family \mathcal{G} of 5-group Cayley graph expanders of degree 6. For graphs $G \in \mathcal{G}$, we can show that the indicator $a(\Phi, G)$ does not vanish, given that Φ is the property of being bipartite. Theorem 2 will then follow by the argument outlined above; the detailed and formal proof is presented in the full version.

In the first step, given a graph $G = \mathcal{C}(\Gamma_i, S_i) \in \mathcal{G}$ we need to establish which fixed-point fractures σ are contained in $\mathcal{L}(\Phi, G)$, that is, for which σ the fractured graph $G\#\sigma$ is bipartite. Since fixed-point fractures $\sigma = (\sigma_v)_{v \in V(G)}$ of G satisfy that all σ_v correspond to one particular partition of S_i , we will ease notation and identify σ with this partition. Using that S_i is a symmetric set of generators of cardinality 6, that is, $S_i = \{g_1, g_2, g_3, g_1^{-1}, g_2^{-1}, g_3^{-1}\}$, we define a graph $\mathcal{H}(\sigma)$ as follows: It has a vertex w^B for each block B of σ , and its set of (multi)edges is given by

$$E(\mathcal{H}(\sigma)) = \left\{ \{w^B, w^{B'}\} : \text{one multiedge for each } g \in \{g_1, g_2, g_3\} \text{ s.t. } g \in B, g^{-1} \in B' \right\}. \quad (5)$$

Note that we see $\mathcal{H}(\sigma)$ as a graph with possible loops and possible multiedges. In particular, the graph $\mathcal{H}(\sigma)$ has precisely 3 edges.

The important property of $\mathcal{H}(\sigma)$, which we prove in the full version, is that $\mathcal{H}(\sigma)$ is bipartite if and only if $G\#\sigma$ is bipartite. Consequently, for Φ being the property of being bipartite, the indicator $a(\Phi, G)$ is given by the following drastically simplified⁶ expression, if considered modulo 5:

$$a(\Phi, G) \equiv \sum_{\sigma: \mathcal{H}(\sigma) \text{ is bipartite}} (-1)^{|\sigma|-1} \cdot (|\sigma| - 1)! \pmod{5}, \quad (6)$$

where the sum is over partitions σ of S_i . We provide the evaluation of the above expression in Table 1 and observe that the result is $-16 \not\equiv 0 \pmod{5}$. Since this argument applies to all members of the family of 5-group Cayley graph expanders, we conclude that the indicator is non-zero infinitely often, which ultimately proves Theorem 2.

The proof of Theorem 1 will follow a comparable technique, but it will require multiple, more involved cases.

Extension to Modular Counting


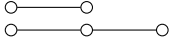



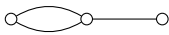

Our understanding of the cancellation behaviour of the indicators $a(\Phi, H)$ modulo p does not only allow us to analyse the complexity of exact counting of k -edge subgraphs satisfying Φ , but also extends to counting k -edge subgraphs modulo p .

To this end, we provide the necessary set-up for parameterized modular counting. In particular, the basis for our intractability results on the modular counting versions is given by the *decision* problem of detecting so-called colour-prescribed homomorphisms, which is known to be hard for graphs of high treewidth due to Marx [31]. Using a version of the Schwartz-Zippel-Lemma due to Williams et al. [46], we are able to reduce an instance I of

⁶ A priori, the formula (4) leads to the version of formula (6) with all summands taken to the power $|V(G)|$. However, since the number of vertices is a power of $p = 5$, by Fermat's little theorem it can be omitted.

84:10 Parameterized (Modular) Counting and Cayley Graph Expanders

■ **Table 1** List of bipartite graphs $\mathcal{H}(\sigma)$ for 3 generators; here we give the isomorphism class of $\mathcal{H}(\sigma)$, the number of partitions σ with the corresponding isomorphism class, the number of blocks of sigma and the total contribution to $a(\Phi, G) \pmod 5$ for the property Φ of being bipartite.

$\mathcal{H}(\sigma)$	No. of σ	$ \sigma $	$(-1)^{ \sigma -1} \cdot (\sigma - 1)!$
	1	6	$-120 \cdot 1$
	12	5	$24 \cdot 12$
	24	4	$-6 \cdot 24$
	8	4	$-6 \cdot 8$
	6	4	$-6 \cdot 6$
	24	3	$2 \cdot 24$
	4	2	$-1 \cdot 4$
Total contribution			$-16 \equiv 4 \pmod 5$

this problem to an instance I' , such that, with high probability, I' has precisely one solution if I has at least one solution, and I' has no solutions if I has no solutions. In a second step, the obtained instance I' can then easily be reduced to the modular counting version for each prime p . Since the first step of this reduction is randomised, we need to assume the randomised Exponential Time Hypothesis for our fine-grained lower bounds.

Afterwards, we prove a variant of the Complexity Monotonicity principle for modular counting in the case of colour-prescribed homomorphisms. As a consequence, our hardness results for modular subgraph counting problems, including Theorem 4, can be proved following the same strategy as outlined in the previous subsection.

Moreover, instead of only presenting intractability results, we investigate whether the expression as a linear combination of homomorphism counts can also be used to achieve improved algorithms for modular subgraph counting problems. And indeed, considering the linear combination (1) modulo 2 for the property Φ of being a path or a cycle, allows us to prove that each graph H with degree at least 5 vanishes in the linear combination, that is, $a_{\Phi, k}(H) = 0$. More precisely, we will prove this for a version of the problem in which two vertices of the k -paths or the k -cycles are already fixed. This must be done to avoid automorphisms of even order, which turns out to be necessary for (1) to be well-defined modulo 2, since some of the coefficients $a_{\Phi, k}(H) = 0$ are of the form $\#\text{Aut}(H)^{-1}$.

The algorithms for counting k -paths and k -cycles modulo 2 turn then out to be very simple: Essentially, we will see that it suffices to guess the two fixed vertices, and thereafter the algorithm evaluates Equation (1) modulo 2, by computing each non-vanishing term using a standard treewidth-based dynamic programming algorithm for counting homomorphisms. Since each graph H whose coefficient survives modulo 2 has degree at most 4, we can rely on known results on the treewidth of bounded degree graphs [19]. Ultimately, this allows us to prove Theorem 5.

Finally, our classification for counting homomorphisms modulo p builds upon the well-established algorithms and reduction sequences used both in the classification for the decision problem [23], as well as in the classification for the exact counting problem [13]. However, the difficulty in proving our classification for counting modulo p is due to graphs H which have high treewidth but admit automorphisms of order p . For those graphs, we can neither rely on an algorithm for exact counting, nor does the known hardness proof transfer.

We solve this problem by considering the p -reduced quotients. Let us denote the function that maps a graph G to the number of homomorphisms from H to G , modulo p , by $\#_p \text{Hom}(H \rightarrow \star)$. We show that for each graph H we have

$$\#_p \text{Hom}(H \rightarrow \star) = \#_p \text{Hom}(H_p^* \rightarrow \star).$$

As a consequence, it suffices to consider the p -reduced quotients for our classification. Since, by definition, those graphs do not admit an automorphism of order p , we are able to show that the known methods for proving classifications for homomorphism problems apply.

Let us conclude by pointing out that, while proving that the p -reduced quotient is uniquely defined up to isomorphism, we also establish a modular variant of Lovász' criterion for graph isomorphism via homomorphism counts (see Chapter 5 in [29]):

► **Lemma 9.** *Let H and H' be graphs, neither of which has an automorphism of order p . Suppose that for all graphs G we have that*

$$\#_p \text{Hom}(H \rightarrow G) = \#_p \text{Hom}(H' \rightarrow G).$$

Then H and H' are isomorphic.

We point out that the previous result can be considered a dual version of [16, Lemma 3.10].

Conclusion and Open Questions

All of our hardness results for modular subgraph counting problems only apply to primes $p \geq 3$ and have, using the randomised Exponential Time Hypothesis as a slightly stronger assumption, the same complexity as their counterparts from exact counting. However, for $p = 2$ the complexity landscape seems different: We obtained an improvement for counting k -cycles and k -paths modulo 2. Moreover, there are known instances of the counting version of the parameterized subgraph isomorphism problem, such as counting k -matchings, where exact counting, as well as counting modulo p for each prime $p \geq 3$ is fixed-parameter **intractable**, while the computation becomes fixed-parameter tractable if done modulo 2 [9, 10].

Since, additionally, many of our hardness proofs do not apply to the case of counting modulo 2, we propose a thorough investigation of the complexity of the parameterized subgraph counting problem modulo 2 as the next step in this line of research. As a starting point, we suggest the problem of counting bipartite k -edge subgraphs modulo 2: While our

proofs extend to counting such subgraphs modulo p for some primes $p \geq 3$, a computer-aided search revealed that, for $p = 2$, our approach *cannot* work for any family of Cayley graph expanders of degree at most 12; details are provided in the full version. Indeed, we conjecture that our methods for proving intractability can be used to show that the problem is intractable for each prime $p > 2$, but not for $p = 2$, which leads to the question of whether this problem might be fixed-parameter tractable.

There are also interesting open questions concerning p -group Cayley graph expanders with low degree. To describe them, fix some prime p and consider the set $D(p) \subseteq \mathbb{Z}_{\geq 0}$ of integers d such that there exists a sequence of finite p -groups Γ_i of orders that tend to infinity, with symmetric generating sets S_i of cardinality $2d$ such that the Cayley graphs $\mathcal{C}(\Gamma_i, S_i)$ form a family of expanders. With any $d \in D(p)$ actually any $d' \geq d$ also lies in $D(p)$, because we can find a uniform bound on the length of a word in S_i to produce a new additional generator for Γ_i , showing $d + 1 \in D(p)$. So the ultimate question is the following:

► **Question 10.** What is the behaviour of the function $p \mapsto d(p) = \min D(p)$?

Since 2-regular graphs are never expanders, we know that $d(p) \geq 2$ for all primes p . Moreover, combining the construction of [36] (for $p = 2$) with our constructions and some further examples that we computed, we obtain the following values and bounds for the function d :

p	$p \in \{2, 3, 5, 7, 11, 13\}$	$17 \leq p \leq 83$	$89 \leq p$
$d(p)$	2	$d(p) \in \{2, 3\}$	$2 \leq d(p) \leq (p + 3)/2$

Based on this experimental evidence we make the following conjecture:

► **Conjecture 11.** For every prime $p \geq 3$ there is a group among the $\Gamma_{p,\alpha,\beta}$ that is 3-generated. In particular, there are p -group Cayley graph expanders of fixed valency $2d$ for all $p \geq 3$ and all $d \geq 3$.

If the conjecture is satisfied, the function d above would be uniformly bounded from above by 3.

References

- 1 Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S. Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249, July 2008. doi:10.1093/bioinformatics/btn163.
- 2 Vikraman Arvind and Venkatesh Raman. Approximation Algorithms for Some Parameterized Counting Problems. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, pages 453–464, 2002. doi:10.1007/3-540-36136-7_40.
- 3 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.
- 4 Andreas Björklund, Holger Dell, and Thore Husfeldt. The Parity of Set Systems Under Random Restrictions with Applications to Exponential Time Problems. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 231–242, 2015. doi:10.1007/978-3-662-47672-7_19.
- 5 Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164, 2018. doi:10.1145/3188745.3188902.

- 6 Andrei A. Bulatov and Amirhossein Kazeminia. Complexity classification of counting graph homomorphisms modulo a prime number. *CoRR*, abs/2106.04086, 2021. arXiv:2106.04086.
- 7 Andrei A. Bulatov and Stanislav Zivný. Approximate counting CSP seen from the other side. *ACM Trans. Comput. Theory*, 12(2):11:1–11:19, 2020. doi:10.1145/3389390.
- 8 Hubie Chen and Stefan Mengel. Counting Answers to Existential Positive Queries: A Complexity Classification. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 – July 01, 2016*, pages 315–326, 2016. doi:10.1145/2902251.2902279.
- 9 Radu Curticapean. Counting Matchings of Size k Is $W[1]$ -Hard. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 352–363. Springer, 2013. doi:10.1007/978-3-642-39206-1_30.
- 10 Radu Curticapean, Holger Dell, and Thore Husfeldt. Modular counting of subgraphs: Matchings, matching-splittable graphs, and paths. In *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal, 2021*. to appear; preprint at arXiv:2107.00629.
- 11 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 210–223, 2017. doi:10.1145/3055399.3055502.
- 12 Radu Curticapean and Dániel Marx. Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139, 2014. doi:10.1109/FOCS.2014.22.
- 13 Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theor. Comput. Sci.*, 329(1-3):315–323, 2004. doi:10.1016/j.tcs.2004.08.008.
- 14 Julian Dörfler, Marc Roth, Johannes Schmitt, and Philip Wellnitz. Counting Induced Subgraphs: An Algebraic Approach to $\#W[1]$ -hardness. In Peter Rossmanith, Pinar Heggenes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPICs*, pages 26:1–26:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.MFCS.2019.26.
- 15 Arnaud Durand and Stefan Mengel. Structural Tractability of Counting of Solutions to Conjunctive Queries. *Theory Comput. Syst.*, 57(4):1202–1249, 2015. doi:10.1007/s00224-014-9543-y.
- 16 John D. Faben and Mark Jerrum. The Complexity of Parity Graph Homomorphism: An Initial Investigation. *Theory Comput.*, 11:35–57, 2015. doi:10.4086/toc.2015.v011a002.
- 17 Jörg Flum and Martin Grohe. The Parameterized Complexity of Counting Problems. *SIAM J. Comput.*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.
- 18 Jacob Focke, Leslie Ann Goldberg, Marc Roth, and Stanislav Zivný. Counting Homomorphisms to K_4 -minor-free Graphs, modulo 2. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10–13, 2021*, pages 2303–2314. SIAM, 2021. doi:10.1137/1.9781611976465.137.
- 19 Fedor V. Fomin, Serge Gaspers, Saket Saurabh, and Alexey A. Stepanov. On Two Techniques of Combining Branching and Treewidth. *Algorithmica*, 54(2):181–207, 2009. doi:10.1007/s00453-007-9133-3.
- 20 Andreas Göbel, Leslie Ann Goldberg, and David Richerby. The complexity of counting homomorphisms to cactus graphs modulo 2. *ACM Trans. Comput. Theory*, 6(4):17:1–17:29, 2014. doi:10.1145/2635825.

- 21 Andreas Göbel, Leslie Ann Goldberg, and David Richerby. Counting Homomorphisms to Square-Free Graphs, Modulo 2. *ACM Trans. Comput. Theory*, 8(3):12:1–12:29, 2016. doi:10.1145/2898441.
- 22 Andreas Göbel, J. A. Gregor Lagodzinski, and Karen Seidel. Counting Homomorphisms to Trees Modulo a Prime. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPICs*, pages 49:1–49:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.MFCS.2018.49.
- 23 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):1:1–1:24, 2007. doi:10.1145/1206035.1206036.
- 24 Heng Guo, Sangxia Huang, Pinyan Lu, and Mingji Xia. The Complexity of Weighted Boolean #CSP Modulo k . In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, volume 9 of *LIPICs*, pages 249–260. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPICs.STACS.2011.249.
- 25 Heng Guo, Pinyan Lu, and Leslie G. Valiant. The Complexity of Symmetric Boolean Parity Holant Problems. *SIAM J. Comput.*, 42(1):324–356, 2013. doi:10.1137/100815530.
- 26 Amirhossein Kazemini and Andrei A. Bulatov. Counting Homomorphisms Modulo a Prime Number. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPICs*, pages 59:1–59:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.MFCS.2019.59.
- 27 J. A. Gregor Lagodzinski, Andreas Göbel, Katrin Casel, and Tobias Friedrich. On Counting (Quantum-)Graph Homomorphisms in Finite Fields. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, to appear*, 2021. Preprint on arXiv:2011.04827.
- 28 Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Efficient Computation of Representative Weight Functions with Applications to Parameterized Counting. Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Alexandria, VA, USA, January 10-13, 2021, to appear.
- 29 László Lovász. *Large Networks and Graph Limits*, volume 60 of *Colloquium Publications*. American Mathematical Society, 2012. URL: <http://www.ams.org/bookstore-getitem/item=COLL-60>.
- 30 A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988. doi:10.1007/BF02126799.
- 31 Dániel Marx. Can You Beat Treewidth? *Theory of Computing*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.
- 32 Dániel Marx. Tractable Hypergraph Properties for Constraint Satisfaction and Conjunctive Queries. *J. ACM*, 60(6):42:1–42:51, 2013. doi:10.1145/2535926.
- 33 Catherine McCartin. Parameterized counting problems. *Ann. Pure Appl. Logic*, 138(1-3):147–182, 2006. doi:10.1016/j.apal.2005.06.010.
- 34 R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002. doi:10.1126/science.298.5594.824.
- 35 Moshe Morgenstern. Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q . *J. Combin. Theory Ser. B*, 62(1):44–62, 1994. doi:10.1006/jctb.1994.1054.
- 36 Norbert Peyerimhoff and Alina Vdovina. Cayley graph expanders and groups of finite width. *J. Pure Appl. Algebra*, 215(11):2780–2788, 2011. doi:10.1016/j.jpaa.2011.03.018.
- 37 Marc Roth and Johannes Schmitt. Counting Induced Subgraphs: A Topological Approach to #W[1]-hardness. *Algorithmica*, 82(8):2267–2291, 2020. doi:10.1007/s00453-020-00676-9.

- 38 Marc Roth, Johannes Schmitt, and Philip Wellnitz. Counting Small Induced Subgraphs Satisfying Monotone Properties. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1356–1367. IEEE, 2020. doi:10.1109/FOCS46700.2020.00128.
- 39 Marc Roth, Johannes Schmitt, and Philip Wellnitz. Detecting and Counting Small Subgraphs, and Evaluating a Parameterized Tutte Polynomial: Lower Bounds via Toroidal Grids and Cayley Graph Expanders. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, to appear*, 2021. Preprint on arXiv:2011.03433.
- 40 Nithi Rungtanapirom, Jakob Stix, and Alina Vdovina. Infinite series of quaternionic 1-vertex cube complexes, the doubling construction, and explicit cubical ramanujan complexes. *International Journal of Algebra and Computation*, 29(6):951–1007, 2019.
- 41 Benjamin Schiller, Sven Jager, Kay Hamacher, and Thorsten Strufe. Stream – A Stream-Based Algorithm for Counting Motifs in Dynamic Graphs. In Adrian-Horia Dediu, Francisco Hernández-Quiroz, Carlos Martín-Vide, and David A. Rosenblueth, editors, *Algorithms for Computational Biology*, pages 53–67, Cham, 2015. Springer International Publishing.
- 42 Jakob Stix and Alina Vdovina. Simply transitive quaternionic lattices of rank 2 over $\mathbb{F}_{q(t)}$ and a non-classical fake quadric. *Mathematical Proceedings of the Cambridge Philosophical Society*, 163(3):453–498, 2017.
- 43 Seinosuke Toda. PP is as Hard as the Polynomial-Time Hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. doi:10.1137/0220053.
- 44 Leslie G. Valiant. The Complexity of Computing the Permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. doi:10.1016/0304-3975(79)90044-6.
- 45 Leslie G. Valiant. Holographic Algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008. doi:10.1137/070682575.
- 46 Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding Four-Node Subgraphs in Triangle Time. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1671–1680, 2015. doi:10.1137/1.9781611973730.111.

A Hierarchy of Nondeterminism

Bader Abu Radi ✉

School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel

Orna Kupferman ✉

School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel

Ofer Leshkowitz ✉

School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel

Abstract

We study three levels in a hierarchy of nondeterminism: A nondeterministic automaton \mathcal{A} is *determinizable by pruning* (DBP) if we can obtain a deterministic automaton equivalent to \mathcal{A} by removing some of its transitions. Then, \mathcal{A} is *good-for-games* (GFG) if its nondeterministic choices can be resolved in a way that only depends on the past. Finally, \mathcal{A} is *semantically deterministic* (SD) if different nondeterministic choices in \mathcal{A} lead to equivalent states. Some applications of automata in formal methods require deterministic automata, yet in fact can use automata with some level of nondeterminism. For example, DBP automata are useful in the analysis of online algorithms, and GFG automata are useful in synthesis and control. For automata on finite words, the three levels in the hierarchy coincide. We study the hierarchy for Büchi, co-Büchi, and weak automata on infinite words. We show that the hierarchy is strict, study the expressive power of the different levels in it, as well as the complexity of deciding the membership of a language in a given level. Finally, we describe a probability-based analysis of the hierarchy, which relates the level of nondeterminism with the probability that a random run on a word in the language is accepting.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases Automata on Infinite Words, Expressive power, Complexity, Games

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.85

1 Introduction

Nondeterminism is a fundamental notion in theoretical computer science. It allows a computing machine to examine several possible actions simultaneously. For automata on finite words, nondeterminism does not increase the expressive power, yet it leads to an exponential succinctness [25].

A prime application of automata theory is specification, verification, and synthesis of reactive systems [29, 15]. Since we care about the on-going behavior of nonterminating systems, the automata run on infinite words. Acceptance in such automata is determined according to the set of states that are visited infinitely often along the run. In *Büchi* automata [9], the acceptance condition is a subset α of states, and a run is accepting iff it visits α infinitely often. Dually, in *co-Büchi* automata, a run is accepting iff it visits α only finitely often. We also consider *weak* automata, which are a special case of both Büchi and co-Büchi automata in which no cycle contains both states in α and states not in α . We use three-letter acronyms in $\{D, N\} \times \{F, B, C, W\} \times \{W\}$ to describe the different classes of automata. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second for the acceptance condition type (finite, Büchi, co-Büchi or weak); and the third indicates that we consider automata on words.

For automata on infinite words, nondeterminism may increase the expressive power and also leads to an exponential succinctness. For example, NBWs are strictly more expressive than DBWs [19], whereas NCWs are as expressive as DCWs [21]. In some applications of the automata-theoretic approach, such as model checking, algorithms can



© Bader Abu Radi, Orna Kupferman, and Ofer Leshkowitz;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 85; pp. 85:1–85:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

be based on nondeterministic automata, whereas in other applications, such as synthesis and control, they cannot. There, the advantages of nondeterminism are lost, and algorithms involve a complicated determinization construction [26] or acrobatics for circumventing determinization [18]. Essentially, the inherent difficulty of using nondeterminism in synthesis and control lies in the fact that each guess of the nondeterministic automaton should accommodate all possible futures.

A study of nondeterministic automata that can resolve their nondeterministic choices in a way that only depends on the past started in [16], where the setting is modeled by means of tree automata for derived languages. It then continued by means of *good for games* (GFG) automata [12].¹ A nondeterministic automaton \mathcal{A} over an alphabet Σ is GFG if there is a strategy g that maps each finite word $u \in \Sigma^*$ to the transition to be taken after u is read; and following g results in accepting all the words in the language of \mathcal{A} . Note that a state q of \mathcal{A} may be reachable via different words, and g may suggest different transitions from q after different words are read. Still, g depends only on the past, namely on the word read so far. Obviously, there exist GFG automata: deterministic ones, or nondeterministic ones that are *determinizable by pruning* (DBP); that is, ones that just add transitions on top of a deterministic automaton. In fact, the GFG automata constructed in [12] are DBP.² Beyond the theoretical interest in DBP automata, they are used for modelling online algorithms: by relating the “unbounded look ahead” of optimal offline algorithms with nondeterminism, and relating the “no look ahead” of online algorithms with determinism, it is possible to reduce questions about the competitive ratio of online algorithms and the memory they require to questions about DBPness [2, 3].

In terms of expressive power, it is shown in [16, 24] that GFG-NXWs, for $X \in \{B, C\}$, are as expressive as DXWs. For automata on finite words, GFG-NFWs are always DBP [16, 22]. For automata on infinite words, GFG-NBW and GFG-NCW need not be DBP [5]. Moreover, the best known determinization construction for GFG-NBW is quadratic, and determinization of GFG-NCW has a tight exponential blow-up [14]. Thus, GFG automata on infinite words are (possibly even exponentially) more succinct than deterministic ones. Further research studies characterization, typeness, complementation, and further constructions and decision procedures for GFG automata [14, 7, 4], as well as an extension of the GFG setting to pushdown ω -automata [20] and to alternating automata [8, 6].

A nondeterministic automaton is *semantically deterministic* (SD, for short) if its non-deterministic choices lead to states with the same language. Thus, for every state q of the automaton and letter $\sigma \in \Sigma$, all the σ -successors of q have the same language. Beyond the fact that semantically determinism is a natural relaxation of determinism, and thus deserves consideration, SD automata naturally arise in the setting of GFG automata. Indeed, though not all GFG automata are DBP, it is not hard to see that they can all be pruned to an SD automaton [14]. Moreover, such a pruning can be done in polynomial time, and so we assume, without loss of generality, that all GFG automata are SD. Thus, SD can be thought also as a natural relaxation of GFG.

Thus, we obtain the following hierarchy, from deterministic to nondeterministic automata, where each level is a special case of the levels to its right.

¹ GFGness is also used in [11] in the framework of cost functions under the name “history-determinism”.

² As explained in [12], the fact that the GFG automata constructed there are DBP does not contradict their usefulness in practice, as their transition relation is simpler than the one of the embodied deterministic automaton and it can be defined symbolically.



For automata on finite words, all levels of the hierarchy coincide in their expressive power. In fact, the three internal levels coincide already in the syntactic sense: every SD-NFW is DBP. Also, given an NFW, deciding whether it is SD, GFG or DBP, can each be done in polynomial time [2].

For Büchi and co-Büchi automata, the picture is less clear, and is the subject of our research. Before we describe our results, let us mention that an orthogonal level of determinism is that of *unambiguous* automata, namely automata that have a single accepting run on each word in their languages. An unambiguous NFW need not be SD, and a DBP-NFW need not be unambiguous. It is known, however, that a GFG unambiguous NFW, NCW, or NBW, is DBP [7].

We study the following aspect and questions about the hierarchy.

Strictness. Recall that not all GFG-NBWs and GFG-NCWs are DBP [5], and examples for this include also SD automata. On the other hand, all GFG-NWWs (in fact, all GFG-NXWs whose language can be recognized by a DWW) are DBP [7]. We show that SD-NXWs need not be GFG for all $X \in \{B, C, W\}$. Of special interest is our result on weak automata, whose properties typically agree with these of automata on finite words. Here, while all SD-NFWs are GFG, this is not the case for SD-NWWs.

Expressive power. It is known that for all $X \in \{B, C, W\}$, GFG-NXWs are as expressive as DXWs. We extend this result to semantic determinism and show that while SD-NXWs need not be GFG, they are not more expressive, thus SD-NXWs are as expressive as DXWs. Since an SD-NXW need not be GFG, this extends the known frontier of nondeterministic Büchi and weak automata that are not more expressive than their deterministic counterpart.

Deciding the determinization level of an automaton. It is already known that deciding the GFGness of a given NXW, for $X \in \{B, C, W\}$, can be done in polynomial time [2, 14, 4]. On the other hand, deciding whether a given NCW is DBP is NP-complete [13]. We complete the picture in three directions. First, we show that NP-completeness of deciding DBPness applies also to NBWs. Second, we show that in both cases, hardness applies even when the given automaton is GFG. Thus, while it took the community some time to get convinced that not all GFG automata are DBP, in fact it is NP-complete to decide whether a given GFG-NBW or GFG-NCW is DBP. Third, we study also the problem of deciding whether a given NXW is SD, and show that it is PSPACE-complete. Note that our results imply that the nondeterminism hierarchy is not monotone with respect to complexity: deciding DBPness, which is closest to determinism, is NP-complete, then GFGness can be checked in polynomial time, and finally SDness is PSPACE-complete. Also, as PSPACE-hardness of checking SDness applies already to NWWs, we get another, even more surprising, difference between weak automata and automata on finite words. Indeed, for NFWs, all the three levels of nondeterminism coincide and SDness can be checked in polynomial time.

A probability-based analysis of the different levels. Consider a nondeterministic automaton \mathcal{A} . We say that \mathcal{A} is *almost-DBP* if we can prune transitions from \mathcal{A} and obtain a deterministic automaton \mathcal{A}' such that the probability of a random word to be in $L(\mathcal{A}) \setminus L(\mathcal{A}')$ is 0. Thus, while \mathcal{A}' need not accept all the words in $L(\mathcal{A})$, it rejects only a negligible fragment of $L(\mathcal{A})$. Clearly, if \mathcal{A} is DBP, then it is almost-DBP. A typical analysis of the performance of an

on-line algorithm compares its performance with that of an off-line algorithm. The notion of almost-DBPness captures cases where the on-line algorithm performs, with probability 1, as good as the offline algorithm. We study the almost-DBPness of GFG and SD automata. We show that while for Büchi (and hence also weak) automata, semantic determinism implies almost-DBPness, thus every SD-NBW is almost-DBP, for co-Büchi automata semantic determinism is not enough, and we need GFGness. Thus, there is an SD-NCW that is not almost-DBP, yet all GFG-NCWs are almost-DBP.

2 Preliminaries

2.1 Automata

For a finite nonempty alphabet Σ , an infinite *word* $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of letters from Σ . A *language* $L \subseteq \Sigma^\omega$ is a set of infinite words. For $i, j \geq 0$, we use $w[1, i]$ to denote the (possibly empty) prefix $\sigma_1 \cdot \sigma_2 \cdots \sigma_i$ of w , use $w[i + 1, j]$ to denote the (possibly empty) infix $\sigma_{i+1} \cdot \sigma_{i+2} \cdots \sigma_j$ of w , and use $w[i + 1, \infty]$ to denote its suffix $\sigma_{i+1} \cdot \sigma_{i+2} \cdots$. We sometimes refer also to languages of finite words, namely subsets of Σ^* . We denote the empty word by ϵ .

A *nondeterministic automaton* over infinite words is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Σ is an alphabet, Q is a finite set of *states*, $q_0 \in Q$ is an *initial state*, $\delta : Q \times \Sigma \rightarrow 2^Q \setminus \emptyset$ is a *transition function*, and α is an *acceptance condition*, to be defined below. For states q and s and a letter $\sigma \in \Sigma$, we say that s is a σ -successor of q if $s \in \delta(q, \sigma)$. Note that \mathcal{A} is *total*, in the sense that it has at least one successor for each state and letter. If $|\delta(q, \sigma)| = 1$ for every state $q \in Q$ and letter $\sigma \in \Sigma$, then \mathcal{A} is *deterministic*.

A *run* of \mathcal{A} on $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states $r = r_0, r_1, r_2, \dots \in Q^\omega$, such that $r_0 = q_0$, and for all $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$. We extend δ to sets of states and finite words in the expected way. Thus, $\delta(S, u)$ is the set of states that \mathcal{A} may reach when it reads the word $u \in \Sigma^*$ from some state in $S \subseteq 2^Q$. Formally, $\delta : 2^Q \times \Sigma^* \rightarrow 2^Q$ is such that for every $S \subseteq 2^Q$, finite word $u \in \Sigma^*$, and letter $\sigma \in \Sigma$, we have that $\delta(S, \epsilon) = S$, $\delta(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma)$, and $\delta(S, u \cdot \sigma) = \delta(\delta(S, u), \sigma)$. The transition function δ induces a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, where for every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, we have that $\langle q, \sigma, s \rangle \in \Delta$ iff $s \in \delta(q, \sigma)$. For a state $q \in Q$ of \mathcal{A} , we define \mathcal{A}^q to be the automaton obtained from \mathcal{A} by setting the initial state to be q . Thus, $\mathcal{A}^q = \langle \Sigma, Q, q, \delta, \alpha \rangle$.

The acceptance condition α determines which runs are “good”. We consider here the *Büchi* and *co-Büchi* acceptance conditions, where $\alpha \subseteq Q$ is a subset of states. We use the terms α -states and $\bar{\alpha}$ -states to refer to states in α and in $Q \setminus \alpha$, respectively. For a run r , let $\text{inf}(r) \subseteq Q$ be the set of states that r traverses infinitely often. Thus, $\text{inf}(r) = \{q \in Q : q = r_i \text{ for infinitely many } i\}$. A run r of a Büchi automaton is *accepting* iff it visits states in α infinitely often, thus $\text{inf}(r) \cap \alpha \neq \emptyset$. Dually, a run r of a co-Büchi automaton is accepting iff it visits states in α only finitely often, thus $\text{inf}(r) \cap \alpha = \emptyset$. A run that is not accepting is *rejecting*. Note that as \mathcal{A} is nondeterministic, it may have several runs on a word w . The word w is accepted by \mathcal{A} if there is an accepting run of \mathcal{A} on w . The language of \mathcal{A} , denoted $L(\mathcal{A})$, is the set of words that \mathcal{A} accepts. Two automata are *equivalent* if their languages are equivalent.

Consider a directed graph $G = \langle V, E \rangle$. A *strongly connected set* in G (SCS, for short) is a set $C \subseteq V$ such that for every two vertices $v, v' \in C$, there is a path from v to v' . A SCS is *maximal* if it is maximal w.r.t containment, that is, for every non-empty set $C' \subseteq V \setminus C$, it holds that $C \cup C'$ is not a SCS. The *maximal strongly connected sets* are also termed *strongly connected components* (SCCs, for short). The *SCC graph* of G is the graph defined over the

SCCs of G , where there is an edge from an SCC C to another SCC C' iff there are two vertices $v \in C$ and $v' \in C'$ with $\langle v, v' \rangle \in E$. A SCC is *ergodic* iff it has no outgoing edges in the SCC graph.

An automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ induces a directed graph $G_{\mathcal{A}} = \langle Q, E \rangle$, where $\langle q, q' \rangle \in E$ iff there is a letter $\sigma \in \Sigma$ such that $\langle q, \sigma, q' \rangle \in \Delta$. The SCCs and SCCs of \mathcal{A} are those of $G_{\mathcal{A}}$. The α -free SCCs of \mathcal{A} are the SCCs of \mathcal{A} that do not contain states from α .

A Büchi automaton \mathcal{A} is *weak* [23] if for each SCC C in $G_{\mathcal{A}}$, either $C \subseteq \alpha$ (in which case we say that C is an accepting SCC) or $C \cap \alpha = \emptyset$ (in which case we say that C is a rejecting SCC). Note that a weak automaton can be viewed as both a Büchi and a co-Büchi automaton, as a run of \mathcal{A} visits α infinitely often, iff it gets trapped in an accepting SCC, iff it visits states in $Q \setminus \alpha$ only finitely often.

We denote the different classes of automata by three-letter acronyms in $\{D, N\} \times \{F, B, C, W\} \times \{W\}$. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second for the acceptance condition type (finite, Büchi, co-Büchi or weak); and the third indicates that we consider automata on words. For example, NBWs are nondeterministic Büchi word automata.

2.2 Probability

Consider the probability space $(\Sigma^\omega, \mathbb{P})$ where each word $w = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots \in \Sigma^\omega$ is drawn by taking the σ_i 's to be independent and identically distributed $\text{Unif}(\Sigma)$. Thus, for all positions $i \geq 1$ and letters $\sigma \in \Sigma$, the probability that σ_i is σ is $\frac{1}{|\Sigma|}$. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ be a deterministic automaton, and let $G_{\mathcal{A}} = \langle Q, E \rangle$ be its induced directed graph. A *random walk on \mathcal{A}* , is a random walk on $G_{\mathcal{A}}$ with the probability matrix $P(q, p) = \frac{|\{\sigma \in \Sigma : \langle q, \sigma, p \rangle \in \Delta\}|}{|\Sigma|}$. It is not hard to see that $\mathbb{P}(L(\mathcal{A}))$ is precisely the probability that a random walk on \mathcal{A} is an accepting run. Note that with probability 1, a random walk on \mathcal{A} reaches an ergodic SCC $C \subseteq Q$, where it visits all states infinitely often. It follows that $\mathbb{P}(L(\mathcal{A}))$ equals the probability that a random walk on \mathcal{A} reaches an ergodic accepting SCC.

2.3 Automata with Some Nondeterminism

Consider a nondeterministic automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. We say that two states $q, s \in Q$ are *equivalent*, denoted $q \sim_{\mathcal{A}} s$, if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$. Then, \mathcal{A} is *semantically deterministic* (SD, for short) if different nondeterministic choices in \mathcal{A} lead to equivalent states. Thus, for every state $q \in Q$ and letter $\sigma \in \Sigma$, all the σ -successors of q are equivalent: for every two states $s, s' \in \delta(q, \sigma)$, we have that $s \sim_{\mathcal{A}} s'$.

An automaton \mathcal{A} is *good for games* (GFG, for short) if its nondeterminism can be resolved based on the past, thus on the prefix of the input word read so far. Formally, \mathcal{A} is GFG if there exists a *strategy* $f : \Sigma^* \rightarrow Q$ such that the following hold:

1. The strategy f is consistent with the transition function. That is, $f(\epsilon) = q_0$, and for every finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $\langle f(u), \sigma, f(u \cdot \sigma) \rangle \in \Delta$.
2. Following f causes \mathcal{A} to accept all the words in its language. That is, for every infinite word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$, if $w \in L(\mathcal{A})$, then the run $f(w[1, 0]), f(w[1, 1]), f(w[1, 2]), \dots$, which we denote by $f(w)$, is an accepting run of \mathcal{A} on w .

We say that the strategy f *witnesses* \mathcal{A} 's GFGness. For an automaton \mathcal{A} , we say that a state q of \mathcal{A} is GFG if \mathcal{A}^q is GFG. Note that every deterministic automaton is GFG. Also, every GFG automaton can be made SD. Indeed, removal of transitions that are not used by a strategy that witnesses \mathcal{A} 's GFGness does not reduce the language of \mathcal{A} and results in an SD automaton. Moreover, by [14, 4], the detection of such transitions can be done in polynomial time.

We say that a nondeterministic automaton \mathcal{A} is *determinizable by pruning* (DBP) if we can remove some of the transitions of \mathcal{A} and get a deterministic automaton \mathcal{A}' that recognizes $L(\mathcal{A})$. We then say that \mathcal{A}' is a *deterministic pruning* of \mathcal{A} . Note that every DBP nondeterministic automaton is GFG. Indeed, the deterministic pruning of \mathcal{A} induces a witness strategy.

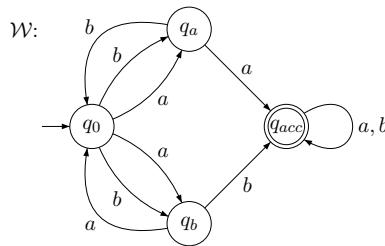
3 The Different Levels and Their Expressive Power

In this section we study syntactic and semantic hierarchies induced by the different levels of nondeterminism. For two classes \mathcal{C}_1 and \mathcal{C}_2 of automata, we use $\mathcal{C}_1 \preceq \mathcal{C}_2$ to indicate that every automaton in \mathcal{C}_1 is also in \mathcal{C}_2 . Accordingly, $\mathcal{C}_1 \prec \mathcal{C}_2$ if $\mathcal{C}_1 \preceq \mathcal{C}_2$ yet there are automata in \mathcal{C}_2 that are not in \mathcal{C}_1 . We first show that the nondeterminism hierarchy is strict, except for all GFG-NWWs being DBP. The latter is not surprising, as all GFG-NFWs are DBP. On the other hand, unlike the case of finite words, we show that not all SD-NWWs are GFG. In fact the result holds already for NWWs that accept co-safety languages, namely all whose states except for an accepting sink are rejecting.

► **Theorem 1** (Syntactic Hierarchy). *For $X \in \{B, C, W\}$, we have that $DXW \prec DBP\text{-}NXW \preceq GFG\text{-}NXW \prec SD\text{-}NXW \prec NXW$. For $X \in \{B, C\}$, the second inequality is strict.*

Proof. By definition, each class is a special case of the one to its right. We prove strictness. It is easy to see that the first and last strict inequalities hold. Indeed, for all $X \in \{B, C, W\}$, consider a nonempty DXW \mathcal{A} , and obtain an NXW \mathcal{B} by adding to \mathcal{A} a σ -transition from the initial state to a new rejecting state, for a letter σ such that \mathcal{A} accepts some word that starts with σ . Then, \mathcal{B} is a DBP-NXW that is not a DXW. Also, as at least one σ -successor of the initial state of \mathcal{A} is not empty, \mathcal{B} is an NXW that is not a SD-NXW.

The relation between DBPness and GFGness has already been studied. It is shown in [5] that GFG-NXW need not be DBP for $X \in \{B, C\}$, and shown in [7] that GFG-NWW are DBP. It is left to relate GFGness and SDness. Consider the NWW \mathcal{W} in Figure 1. It is not hard to check that \mathcal{W} is indeed weak, it is SD, as all its states recognize the language $\{a, b\}^\omega$, yet is not GFG, as every strategy has a word with which it does not reach q_{acc} – a word that forces each visit in q_a and q_b to be followed by a visit in q_0 .



■ **Figure 1** An SD-NWW that is not GFG.

Hence GFG-NWW \prec SD-NWW. As weak automata are a special case of Büchi and co-Büchi, strictness for them follows. ◀

We continue to study expressive power. Now, for two classes \mathcal{C}_1 and \mathcal{C}_2 of automata, we say that \mathcal{C}_1 is less expressive than \mathcal{C}_2 , denoted $\mathcal{C}_1 \leq \mathcal{C}_2$, if every automaton in \mathcal{C}_1 has an equivalent automaton in \mathcal{C}_2 . Since NCW=DCW, we expect the hierarchy to be strict only in the cases of Büchi and weak automata. As we now show, however, semantically deterministic automata are not more expressive than deterministic ones also in the case of Büchi and weak automata.

► **Theorem 2** (Expressiveness Hierarchy). *For $X \in \{B, W\}$, we have that $DXW = DBP\text{-}NXW = GFG\text{-}NXW = SD\text{-}NXW < NXW$.*

Proof. In [16, 14], the authors suggest variants of the subset construction that determinize GFG-NBWs. As we argue below, the construction in [14] is correct also when applied to SD-NBWs. Moreover, it preserves weakness. Thus, $DBW = SD\text{-}NBW$ and $DWW = SD\text{-}NWW$. Also, the last inequality follows from the fact $DBW < NBW$ and $DWW < NWW$ [19].

Given an NBW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, the DBW generated in [14]³ is $\mathcal{A}' = \langle \Sigma, Q', q'_0, \delta', \alpha' \rangle$, where $Q' = 2^Q$, $q'_0 = \{q_0\}$, $\alpha' = \{S \in 2^Q : S \subseteq \alpha\}$, and the transition function δ' is defined for every subset $S \in 2^Q$ and letter $\sigma \in \Sigma$ as follows. If $\delta(S, \sigma) \cap \alpha = \emptyset$, then $\delta'(S, \sigma) = \delta(S, \sigma)$. Otherwise, namely if $\delta(S, \sigma) \cap \alpha \neq \emptyset$, then $\delta'(S, \sigma) = \delta(S, \sigma) \cap \alpha$.

The key observation about the correctness of the construction is that when \mathcal{A} is an SD-NBW, then for all reachable states S of \mathcal{A}' , we have that $q \sim_{\mathcal{A}} q'$ for all states $q, q' \in S$. Indeed, if \mathcal{A} is SD, then for every two states $q, q' \in Q$, letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, s \rangle, \langle q', \sigma, s' \rangle \in \Delta$, if $q \sim_{\mathcal{A}} q'$, then $s \sim_{\mathcal{A}} s'$. Also, by the definition of δ' , every reachable state S of \mathcal{A}' contains only α -states or only $\bar{\alpha}$ -states. As we formally prove in Appendix A, these properties guarantee that indeed $L(\mathcal{A}') = L(\mathcal{A})$ and that weakness of \mathcal{A} is maintained in \mathcal{A}' . ◀

4 Deciding the Nondeterminism Level of an Automaton

In this section we study the complexity of the problem of deciding the nondeterminism level of a given automaton. Note we refer here to the syntactic class (e.g., deciding whether a given NBW is GFG) and not to the semantic one (e.g., deciding whether a given NBW has an equivalent GFG-NBW). Indeed, by Theorem 2, the latter boils down to deciding whether the language of a given NXW can be recognized by a DXW, which is well known: the answer is always “yes” for an NCW, and the problem is PSPACE-complete for NBWs and NWWs [17].⁴

Our results are summarized in Table 1. The entries there describe both the case in which the given automaton is a general NXW, and the case in which the given automaton is an NXW that belongs to a level, that is one level to the right of the questioned one (for example, deciding DBPness of a GFG automaton). In fact, the complexity of the two cases coincide, with one exception: deciding whether a given NWW is DBP, which is PTIME in general, and is $O(1)$ when the given NWW is GFG, in which case the answer is always “yes”.

► **Theorem 3.** *Deciding whether an NXW is semantically deterministic is PSPACE-complete, for $X \in \{B, C, W\}$.*

Proof. Membership in PSPACE is easy, as we check SDness by polynomially many checks of language equivalence. Formally, given an NXW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, a PSPACE algorithm goes over all states $q \in Q$, letters σ , and σ -successors s and s' of q , and checks that $s \sim_{\mathcal{A}} s'$. Since language equivalence can be checked in PSPACE [28] and there are polynomially many checks to perform, we are done.

³ The construction in [14] assumes automata with transition-based acceptance, and (regardless of this) is slightly different: when α is visited, \mathcal{A}' continues with a single state from the set of successors. The key point, however, is the same: \mathcal{A} being SD enables \mathcal{A}' to maintain only subsets of states, rather than Safra trees, which makes determinization much easier.

⁴ The proof in [17] is for NBWs, yet the arguments there apply also for weak automata.

■ **Table 1** Deciding the level of an NXW. The results are valid also in the case the given NXW is one level to the right in the nondeterminism hierarchy. Two exceptions are the cases of deciding the DBPness of a GFG NCW and GFG NWW, where the results are specified in ().

	DBP	GFG	SD
NBW	NP-complete Th. 4	PTIME [4]	PSPACE-complete Th. 3
NCW	NP-complete [13] (Th. 8)	PTIME [14]	PSPACE-complete Th. 3
NWW	PTIME ($O(1)$) [14, 4]([7])	PTIME [14, 4]	PSPACE-complete Th. 3

Proving PSPACE-hardness, we do a reduction from polynomial-space Turing machines. Given a Turing machine T with space complexity $s : \mathbb{N} \rightarrow \mathbb{N}$, we construct in time polynomial in $|T|$ and $s(0)$, an NWW \mathcal{A} of size linear in T and $s(0)$, such that \mathcal{A} is SD iff T accepts the empty tape⁵. Clearly, this implies a lower bound also for NBWs and NCWs. Let $n_0 = s(0)$. Thus, each configuration in the computation of T on the empty tape uses at most n_0 cells. We assume, without loss of generality, that once T reaches a final (accepting or rejecting) state, it erases the tape, moves with its reading head to the leftmost cell, and moves to the initial state. Thus, all computations of T are infinite and after visiting a final configuration for the first time, they consists of repeating the same finite computation on the empty tape that uses n_0 tape cells.

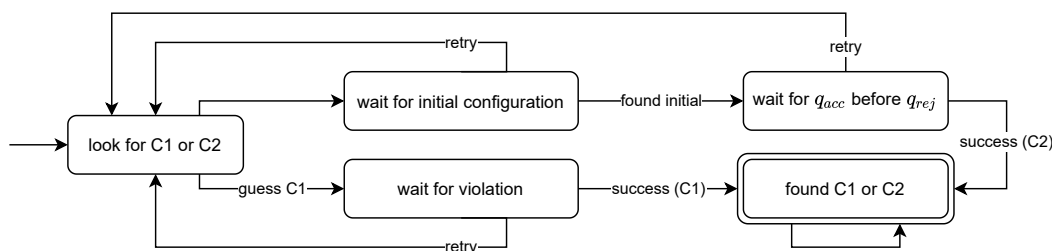
We define \mathcal{A} so that it accepts a word w iff (C1) w is not a suffix of an encoding of a legal computation of T that uses at most n_0 cells, or (C2) w includes an encoding of the initial configuration of T on the empty tape and the final configuration after it, is accepting.

It is not hard to see that if T accepts the empty tape, then \mathcal{A} is universal (that is, accepts all words). Indeed, each word w is either not a suffix of an encoding of a legal computation of T that uses at most n_0 cells, in which case w is accepted thanks to C1. Otherwise, the encoding of the computation of T on the empty tape is a subword of w , in which case w eventually includes an encoding of the initial configuration of T on the empty tape, and the final configuration after it is accepting, and thus w is accepted thanks to C2. Also, if T rejects the empty tape, then \mathcal{A} rejects the word that encodes the computation of T on the empty tape. Indeed, C1 is not satisfied, and since every encoding of the initial configuration is followed by an encoding of the rejecting computation of T , the final configuration after it is rejecting, and so C2 is not satisfied too.

In order to define \mathcal{A} so that it is SD iff T accepts the empty tape, we define all its states to be universal iff T accepts the empty tape. Intuitively, we do it by letting \mathcal{A} guess and check the existence of an infix that witnesses satisfaction of C1 or C2, and also let it, at each point of its operation, go back to the initial state, where it can guess again. Note that when T accepts the empty tape, all the suffixes of a word w satisfy C1 or C2. Thus, \mathcal{A} making a bad guess does not prevent it from later branching into an accepting run.

⁵ This is sufficient, as one can define a generic reduction from every language L in PSPACE as follows. Let T_L be a Turing machine that decides L in polynomial space $f(n)$. On input w for the reduction, the reduction considers the machine T_w that on every input, first erases the tape, writes w on its tape, and then runs as T_L on w . Then, the reduction outputs an automaton \mathcal{A} , such that T_w accepts the empty tape iff \mathcal{A} is SD. Note that the space complexity of T_w is $s(n) = \max(n, f(|w|))$, and that w is in L iff T_w accepts the empty tape. Since \mathcal{A} is constructed in time polynomial in $s(0) = f(|w|)$ and $|T_w| = \text{poly}(|w|)$, it follows that the reduction is polynomial in $|w|$.

We now describe the operation of \mathcal{A} in more detail (see Figure 2).



■ **Figure 2** The structure of the NWW constructed in Theorem 3.

In its initial state, \mathcal{A} guesses which of $C1$ and $C2$ is satisfied. In case \mathcal{A} guesses that $C1$ is satisfied, it guesses the place in which w includes a violation of the encoding. As we detail in Appendix B, this amounts to guessing a violation of the transition function of T : in each step, \mathcal{A} may guess that the next three letters encode a position in a configuration and the letter to come n_0 letters later, namely at the same position in the successive configuration, is different from the one that should appear in a legal encoding of two successive configurations. If a violation is detected, \mathcal{A} moves to an accepting sink. Otherwise, \mathcal{A} returns to the initial state and w gets another chance to be accepted. In case \mathcal{A} guesses that $C2$ is satisfied, it guesses the place in which w encodes an initial configuration. If \mathcal{A} guesses a position of an initial configuration, but the guess fails, then \mathcal{A} goes back to the initial state. If the guess succeeds, \mathcal{A} waits for an accepting configuration of T . If an accepting configuration arrives before a rejecting one, then \mathcal{A} moves to an accepting sink. Otherwise, if a rejecting configuration arrives before an accepting one, then \mathcal{A} returns to the initial state. Also, whenever \mathcal{A} waits to witness some behavior, namely, waits to guess a position of an initial state, waits to guess a position of a violation, or waits to see a final configuration, it may nondeterministically, upon reading the next letter, return to the initial state. It is not hard to see that \mathcal{A} can be defined in size linear in T and n_0 . As the only accepting states of \mathcal{A} is the accepting sink, it is clearly weak, and in fact describes a co-safety language.

We prove that T accepts the empty tape iff \mathcal{A} is SD. First, if T rejects the empty tape, then \mathcal{A} is not SD. To see this, consider the word w_ε that encodes the computation of T on the empty tape, and let w'_ε be a word that is obtained from w_ε by making a single violation in the first letter. That is, $w'_\varepsilon[2, \infty] = w_\varepsilon[2, \infty]$, and $w'_\varepsilon[1, 1] \neq w_\varepsilon[1, 1]$. Note that $w'_\varepsilon \in L(\mathcal{A})$ since it has a violation. Note also that any proper suffix of w'_ε encodes a suffix of a computation of T that uses at most n_0 tape cells and does not have of a final accepting configuration, and hence is not in $L(\mathcal{A})$. Consequently, the word w'_ε can be accepted by \mathcal{A} only by guessing a violation that is caused by the first letter. In particular, if we guess to wait for the initial configuration upon reading the first letter, then we cannot branch to an accepting run. This shows that \mathcal{A} is not SD. For the other direction, we show that if T accepts the empty tape, then all the states of \mathcal{A} are universal. First, note that each infinite word w is either not a suffix of a legal encoding of a computation of T that uses at most n_0 tape cells, in which case it is in the language of \mathcal{A} by $C1$, or it is a suffix of a legal encoding of a computation that uses only n_0 tapes cells, and is eventually an encoding of the computation of T on the empty tape, in which case, as T accepts the empty tape, w is in the language of \mathcal{A} according to $C2$. Thus, the initial state of \mathcal{A} is universal. Now by the definition of \mathcal{A} , for every infinite word w and for all states q of \mathcal{A} that are not the accepting sink, there is a path from q to the initial state that is labeled by a prefix of w . Thus, the language of all states is universal, and they are all equivalent. This clearly implies that \mathcal{A} is SD.

Thus, we conclude that T accepts the empty tape iff \mathcal{A} is SD. In Appendix B, we give the full technical details of the construction of \mathcal{A} . ◀

► **Theorem 4.** *The problem of deciding whether a given GFG-NBW is DBP is NP-complete.*

Proof. For membership in NP, observe we can check that a witness deterministic pruning \mathcal{A}' is equivalent to \mathcal{A} by checking whether $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. Since \mathcal{A}' is deterministic, the latter can be checked in polynomial time. For NP-hardness, we describe a parsimonious polynomial time reduction from SAT. That is, given a CNF formula φ , we construct a GFG-NBW \mathcal{A}_φ such that there is a bijection between assignments to the variables of φ and DBWs embodied in \mathcal{A}_φ , and an assignment satisfies φ iff its corresponding embodied DBW is equivalent to \mathcal{A}_φ . In particular, φ is satisfiable iff \mathcal{A}_φ is DBP.

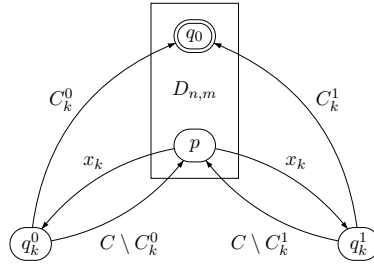
Consider a SAT instance φ over the variable set $X = \{x_1, \dots, x_n\}$ and with $m \geq 1$ clauses $C = \{c_1, \dots, c_m\}$. For $n \geq 1$, let $[n] = \{1, 2, \dots, n\}$. For a variable $x_k \in X$, let $C_k^0 \subseteq C$ be the set of clauses in which x_k appears negatively, and let $C_k^1 \subseteq C$ be the set of clauses in which x_k appears positively. For example, if $c_1 = x_1 \vee \neg x_2 \vee x_3$, then c_1 is in C_1^1 , C_2^0 , and C_3^1 . Assume that all clauses depend on at least two different variables (that is, no clause is a tautology or forces an assignment to a single variable). Let $\Sigma_{n,m} = X \cup C$, and let $R_{n,m} = (X \cdot C)^* \cdot \{x_1 \cdot c_j \cdot x_2 \cdot c_j \cdots x_n \cdot c_j : j \in [m]\} \subseteq \Sigma_{n,m}^*$. We construct a GFG-NBW \mathcal{A}_φ that recognizes $L_{n,m} = (R_{n,m})^\omega$, and is DBP iff φ is satisfiable.

Let $D_{n,m}$ be a DFW that recognizes $R_{n,m}$ with $O(n \cdot m)$ states, a single accepting state p , and an initial state q_0 that is visited only once in all runs. For example, we can define $D_{n,m} = \langle \Sigma_{n,m}, Q_{n,m}, q_0, \delta_{n,m}, \{p\} \rangle$ as follows: from q_0 , the DFW expects to read only words in $(X \cdot C)^*$ – upon a violation of this pattern, it goes to a rejecting sink. Now, if the pattern is respected, then with $X \setminus \{x_1\}$, the DFW goes to two states where it loops with $C \cdot (X \setminus \{x_1\})$ and, upon reading x_1 from all states that expect to see letters in X , it branches with each c_j , for all $j \in [m]$, to a path where it hopes to detect an $x_2 \cdot c_j \cdots x_n \cdot c_j$ suffix. If the detection is completed successfully, it goes to the accepting state p . Otherwise, it returns to the two-state loop.

Now, we define $\mathcal{A}_\varphi = \langle \Sigma_{n,m}, Q_\varphi, p, \delta_\varphi, \{q_0\} \rangle$, where $Q_\varphi = Q_{n,m} \cup \{q_k^i : (i, k) \in \{0, 1\} \times [n]\}$. The idea behind \mathcal{A}_φ is as follows. From state p (that is, the accepting state of $D_{n,m}$, which is now the initial state of \mathcal{A}_φ), the NBW \mathcal{A}_φ expects to read a letter in X . When it reads x_k , for $1 \leq k \leq n$, it nondeterministically branches to the states q_k^0 and q_k^1 . Intuitively, when it branches to q_k^0 , it guesses that the clause that comes next is one that is satisfied when $x_k = 0$, namely a clause in C_k^0 . Likewise, when it branches to q_k^1 , it guesses that the clause that comes next is one that is satisfied when $x_k = 1$, namely a clause in C_k^1 . When the guess is successful, \mathcal{A}_φ moves to the α -state q_0 . When the guess is not successful, it returns to p . Implementing the above intuition, transitions from the states $Q_{n,m} \setminus \{p\}$ are inherited from $D_{n,m}$, and transitions from the states in $\{q_k^i : (i, k) \in \{0, 1\} \times [n]\} \cup \{p\}$ are defined as follows (see also Figure 3).

- For all $k \in [n]$, we have that $\delta_\varphi(p, x_k) = \{q_k^0, q_k^1\}$.
- For all $k \in [n]$, $i \in \{0, 1\}$, and $j \in [m]$, if $c_j \in C_k^i$, then $\delta_\varphi(q_k^i, c_j) = \{q_0\}$. Otherwise, $\delta_\varphi(q_k^i, c_j) = \{p\}$. For example, if $c_1 = x_1 \vee \neg x_2 \vee x_3$, then $\delta_\varphi(q_2^0, c_1) = \{q_0\}$ and $\delta_\varphi(q_2^1, c_1) = \{p\}$.

Note that p is the only nondeterministic state of \mathcal{A}_φ and that for every deterministic pruning of \mathcal{A}_φ , all the words in $(X \cdot C)^\omega$ have an infinite run in the pruned automaton. This run, however, may eventually loop in $\{p\} \cup \{q_k^0, q_k^1 : k \in [n]\}$. Note also, that for readability purposes, the automaton \mathcal{A}_φ is not total. Specifically, the states of \mathcal{A}_φ are partitioned into



■ **Figure 3** The transitions to and from the states q_k^0 and q_k^1 in \mathcal{A}_φ .

states that expect to see letters in X and states that expect to see letters in C . In particular, all infinite paths in \mathcal{A}_φ are labeled by words in $(X \cdot C)^\omega$. Thus, when defining a GFG strategy g for \mathcal{A}_φ , we only need to define g on prefixes in $(X \cdot C)^* \cup (X \cdot C)^* \cdot X$.

In the following propositions, we prove that \mathcal{A}_φ is a GFG NBW recognizing $L_{n,m}$, and that \mathcal{A}_φ is DBP iff φ is satisfiable. ◀

► **Proposition 5.** $L(\mathcal{A}_\varphi) \subseteq L_{n,m}$.

Proof. As already mentioned, all infinite paths of \mathcal{A}_φ , accepting or rejecting, are labeled by words in $(X \cdot C)^\omega$. Further, any accepting run of \mathcal{A}_φ has infinitely many sub-runs that are accepting finite runs of $D_{n,m}$. Since $L_{n,m} = (R_{n,m})^\omega = (X \cdot C)^\omega \cap (\infty R_{n,m})$, it follows that $L(\mathcal{A}_\varphi) \subseteq L_{n,m}$. ◀

► **Proposition 6.** *There exists a strategy $g : \Sigma^* \rightarrow Q_\varphi$ for \mathcal{A}_φ that accepts all words in $L_{n,m}$. Formally, for all $w \in L_{n,m}$, the run $g(w) = g(w[1, 0]), g(w[1, 1]), g(w[1, 2]), \dots$, is an accepting run of \mathcal{A}_φ on w .*

Proof. The definition of $L_{n,m}$ is such that when reading a prefix that ends with a subword of the form $x_1 \cdot c_j$, for some $j \in [m]$, then we can guess that the word continues with $x_2 \cdot c_j \cdot x_3 \cdot c_j \cdots x_n \cdot c_j$; thus that c_j is the clause that is going to repeat. Therefore, when we are at state p after reading a word that ended with $x_1 \cdot c_j$, and we read x_2 , it is a good GFG strategy to move to a state q_2^i such that the assignment $x_2 = i$ satisfies c_j (if such $i \in \{0, 1\}$ exists; otherwise the strategy can choose arbitrary between q_2^0 and q_2^1), and if the run gets back to p , the strategy continues with assignments that hope to satisfy c_j , until the run gets to q_0 or another occurrence of x_1 is detected. Note that while it is not guaranteed that for all $k \in [n]$ there is $i \in \{0, 1\}$ such that the assignment $x_k = i$ satisfies c_j , it is guaranteed that such an i exists for at least two different k 's (we assume that all clauses depend on at least two variables). Thus, even though we a priori miss an opportunity to satisfy c_j with an assignment to x_1 , it is guaranteed that there is another $2 \leq k \leq n$ such that c_j can be satisfied by x_k .

We define g inductively as follows. Recall that \mathcal{A}_φ is nondeterministic only in the state p , and so in all other states, the strategy g follows the only possible transition. First, for all $k \in [n]$, we define $g(x_k) = q_k^0$. Let $v \in (X \cdot C)^* \cdot X$, be such that g has already been defined on v and let $j \in [m]$. Since $v \notin (X \cdot C)^*$, we have that $g(v) \neq p$ and so $g(v \cdot c_j)$ is uniquely defined. We continue and define g on $u = v \cdot c_j \cdot x_k$, for all $k \in [n]$. If $g(v \cdot c_j) \neq p$, then $g(u)$ is uniquely defined. Otherwise, $g(v \cdot c_j) = p$ and we define $g(u)$ as follows,

- If $k = 1$, then we define $g(u) = q_1^0$.
- If $k > 1$ and x_k participates in c_j , then we define $g(u) = q_k^i$, where $i \in \{0, 1\}$ is minimal with $c_j \in C_k^i$. That is, i is the minimal assignment to x_k that satisfies c_j .
- If $k > 1$ and x_k does not participate in c_j , then the value of c_j is not affected by the assignment to x_k , and in that case we define $g(u) = q_k^0$.

The reason for the distinction between the cases $k = 1$ and $k > 1$ is that when we see a finite word that ended with $c_j \cdot x_1$, then there is no special reason to hope that the next letter is going to be c_j . This is in contrast, for example, to the case we have seen a word that ends with $c_{j'} \cdot x_1 \cdot c_j \cdot x_2$, where it is worthwhile to guess we are about to see c_j as the next letter.

By the definition of g , it is consistent with Δ_φ . In Appendix C we formally prove that g is a winning GFG strategy for \mathcal{A}_φ . Namely, that for all $w \in L_{n,m}$, the run $g(w)$ on w , generated by g is accepting. ◀

We now examine the relation between prunings of \mathcal{A}_φ and assignments to φ . Consider an assignment $i_1, \dots, i_n \in \{0, 1\}$, for X . I.e., $x_k = i_k$ for all $k \in [n]$. Then a possible memoryless GFG strategy, is to always move from p to $q_k^{i_k}$ when reading x_k . This in fact, describes a one to one correspondence, between assignments and prunings of \mathcal{A}_φ . Assume that the assignment $i_k \in \{0, 1\}$, for $k \in [n]$, satisfies φ , then the corresponding pruning recognizes $L_{n,m}$. Indeed, instead of trying to satisfy the last read clause c_j , we may ignore this extra information, and rely on the fact that one of the assignments $x_k = i_k$ is going to satisfy c_j . In other words, the satisfiability of φ allows us to ignore the history and still accept all words in $L_{n,m}$, which makes \mathcal{A}_φ DBP. On the other hand, if an assignment does not satisfy some clause c_j , then the corresponding pruning will fail to accept the word $(x_1 \cdot c_j \cdots x_n \cdot c_j)^\omega$, which shows that if φ is not satisfiable then \mathcal{A}_φ is not DBP. In Appendix C we formally prove that there is a one to one correspondence between prunings of \mathcal{A}_φ and assignments to φ , and that an assignment satisfies φ iff the corresponding pruning recognizes $L_{n,m}$, implying Proposition 7.

► **Proposition 7.** *The formula φ is satisfiable iff the GFG-NBW \mathcal{A}_φ is DBP.*

We continue to co-Büchi automata. In [13], the authors prove that deciding the DBPness of a given NCW is NP-complete. For the lower bound, they describe a reduction from the *Hamiltonian-cycle* problem. Essentially, given a connected graph $G = \langle [n], E \rangle$, the reduction outputs an NCW \mathcal{A}_G over the alphabet $[n]$ that is obtained from G by adding self loops to all vertices, labelling the loop at a vertex i by the letter i , and labelling the edges from vertex i to all its neighbours in G by every letter $j \neq i$. Then, the co-Büchi condition requires a run to eventually get stuck at a self-loop⁶. Accordingly, $L(\mathcal{A}_G) = [n]^* \cdot \bigcup_{i \in [n]} i^\omega$.

It is not hard to see that \mathcal{A}_G is GFG. Indeed, a GFG strategy can decide to which neighbour of i to proceed with a letter $j \neq i$ by following a cycle c that traverses all the vertices of the graph G . Since when we read $j \neq i$ at vertex i we move to a neighbour state, then by following the cycle c upon reading i^ω , we eventually reach the vertex i and get stuck at the i -labeled loop. Thus, the NP-hardness result of [13] apply already for GFG-NCWs, and we can conclude with the following.

► **Theorem 8.** *The problem of deciding whether a given GFG-NCW is DBP is NP-complete.*

5 A probability-Based Analysis of the Different Levels

Consider a nondeterministic automaton \mathcal{A} . We say that \mathcal{A} is *almost-DBP* if there is a deterministic pruning \mathcal{A}' of \mathcal{A} such that $\mathbb{P}(L(\mathcal{A}) \setminus L(\mathcal{A}')) = 0$. Thus, while \mathcal{A}' need not accept all the words accepted by \mathcal{A} , it rejects only a negligible set of words in $L(\mathcal{A})$. Clearly, if \mathcal{A} is DBP, then it is almost-DBP. In this section we study the almost-DBPness of GFG

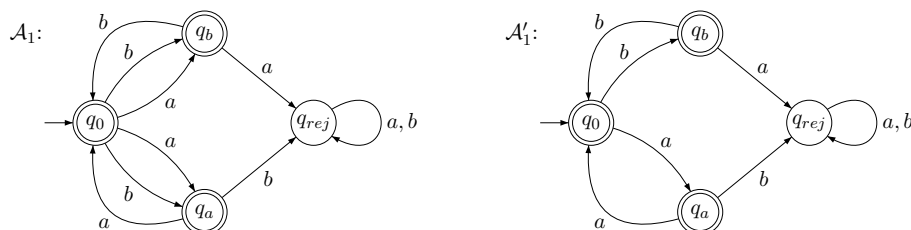
⁶ The exact reduction is more complicated and involves an additional letter $\#$ that forces each deterministic pruning of \mathcal{A}_G to proceed to the same neighbour of i upon reading a letter $j \neq i$ from the vertex i .

and SD automata. We show that while for Büchi (and hence also weak) automata, semantic determinism implies almost-DBPness, thus every SD-NBW is almost-DBP, for co-Büchi automata semantic determinism is not enough, and we need GFGness. Thus, there is an SD-NCW that is not almost-DBP, yet all GFG-NCWs are almost-DBP.

We first show that, unsurprisingly, not all NBWs are almost-DBP.

► **Theorem 9.** *There is an NBW that is not almost-DBP.*

Proof. Consider the NBW \mathcal{A}_1 in Figure 4. It is not hard to see that $L(\mathcal{A}_1) = \{a, b\}^\omega$, and so $\mathbb{P}(L(\mathcal{A}_1)) = 1$. Moreover, every deterministic pruning of \mathcal{A}_1 is such that q_{rej} is reachable from all states, which implies that $\{q_{rej}\}$ is the only ergodic SCC of any pruning. Since $\{q_{rej}\}$ is α -free, it follows that every deterministic pruning of \mathcal{A}_1 recognizes a language of measure zero, and hence \mathcal{A}_1 is not almost-DBP. As an example, consider the deterministic pruning \mathcal{A}'_1 described on the right hand side of Figure 4. The only ergodic SCC of \mathcal{A}'_1 is α -free, and as such $\mathbb{P}(L(\mathcal{A}'_1)) = 0$. ◀



■ **Figure 4** An NBW that is not almost-DBP.

We continue to the positive result about Büchi automata. Consider an NBW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. We define a *simple stochastic Büchi game* $\mathcal{G}_{\mathcal{A}}$ as follows.⁷ The game is played between Random and Eve. The positions of Random are Q , these of Eve are $Q \times \Sigma$. The game starts from position q_0 . A round in the game starts at some position $q \in Q$ and proceeds as follows.

1. Random picks a letter $\sigma \in \Sigma$ uniformly, and the game moves to position (q, σ) .
2. Eve picks a transition $(q, \sigma, p) \in \Delta$, and the game moves to position p .

A probabilistic strategy for Eve is $f : (Q \times \Sigma)^+ \rightarrow [0, 1]^Q$, where for all histories $x \in (Q \times \Sigma)^*$ and positions of Eve $(q, \sigma) \in Q \times \Sigma$, the function $d = f(x \cdot (q, \sigma)) : Q \rightarrow [0, 1]$, is a distribution on Q such that $d(p) \neq 0$ implies that $p \in \delta(q, \sigma)$. As usual, we say that a strategy f is *memoryless*, if it depends only on the current position, thus for all histories $x, y \in (Q \times \Sigma)^*$ and positions of Eve $(q, \sigma) \in Q \times \Sigma$, it holds that $f(x \cdot (q, \sigma)) = f(y \cdot (q, \sigma))$. A strategy for Eve is *pure* if for all histories $x \in (Q \times \Sigma)^*$ and positions of Eve $(q, \sigma) \in Q \times \Sigma$, there is a position $p \in \delta(q, \sigma)$ such that $f(x \cdot (q, \sigma))(p) = 1$. When Eve plays according to a strategy f , the outcome of the game can be viewed as a run $r_f = q_f^0, q_f^1, q_f^2, \dots$ in \mathcal{A} , over a random word $w_f \in \Sigma^\omega$. (The word that is generated in a play is independent of the strategy of Eve, but we use the notion w_f to emphasize that we are considering the word that is generated in a play where Eve plays according to f).

Let $Q_{rej} = \{q \in Q : \mathbb{P}(L(\mathcal{A}^q)) = 0\}$. The outcome r_f of the game is *winning* for Eve iff r_f is accepting, or r_f visits Q_{rej} . Note that for all positions $q \in Q_{rej}$ and $p \in Q$, if p is reachable from q , then $p \in Q_{rej}$. Hence, the winning condition can be defined by the Büchi

⁷ In [10] these games are called simple $1\frac{1}{2}$ -player games with Büchi winning objectives and almost-sure winning criterion.

objective $\alpha \cup Q_{rej}$. Note that r_f is winning for Eve iff $\text{inf}(r_f) \subseteq Q_{rej}$ or $\text{inf}(r_f) \cap \alpha \neq \emptyset$. We say that f is an *almost-sure winning* strategy, if r_f is winning for Eve with probability 1, and Eve *almost-sure wins* in $\mathcal{G}_{\mathcal{A}}$ if she has an almost-sure winning strategy.

► **Theorem 10.** *All SD-NBWs are almost-DBP.*

Proof. We first show that Eve has a probabilistic strategy to win $\mathcal{G}_{\mathcal{A}}$ with probability 1, even without assuming that \mathcal{A} is semantically deterministic. Consider the probabilistic strategy g where from (q, σ) , Eve picks one of the σ -successors of q uniformly by random. Note that this strategy is memoryless, and hence the outcome of the game can be thought as a random walk in \mathcal{A} that starts at q_0 and gives positive probabilities to all transitions. Thus, with probability 1, the run r_g is going to reach an ergodic SCC of \mathcal{A} and visit all its states. If the run r_g reaches an α -free ergodic SCC, then $\text{inf}(r_g) \subseteq Q_{rej}$, and hence r_g is then winning for Eve. Otherwise, r_g reaches a non α -free ergodic SCC, and with probability 1, it visits all the states in that SCC. Thus, with probability 1, we have $\text{inf}(r_g) \cap \alpha \neq \emptyset$, and r_g is winning for Eve. Overall, Eve wins $\mathcal{G}_{\mathcal{A}}$ with probability 1 when playing according to g .

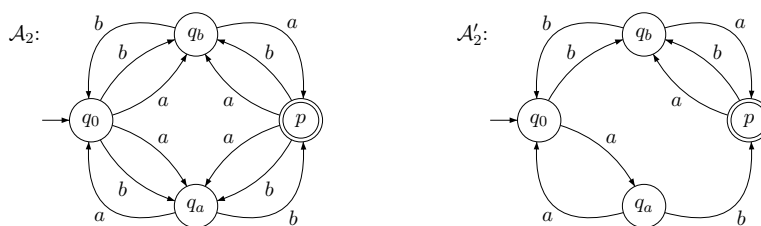
Hence, by pure memoryless determinacy of simple stochastic parity games [10], we may consider a pure memoryless winning strategy f for Eve in $\mathcal{G}_{\mathcal{A}}$. We say that r_f is *correct* if $w_f \in L(\mathcal{A})$ implies that r_f is accepting. Note that $w_f \notin L(\mathcal{A})$ always implies that r_f is rejecting. We show that if \mathcal{A} is SD, then r_f is correct with probability 1, where f is a pure memoryless winning strategy for Eve. Since f is pure memoryless, it induces a pruning of \mathcal{A} . Denote this pruning by \mathcal{A}^f . We may think of r_f as a random walk in \mathcal{A}^f . With probability 1, the walk r_f reaches an ergodic SCC C of \mathcal{A}^f , and visits all its states. Since f is a winning strategy, we know that $C \subseteq Q_{rej}$ or $C \cap \alpha \neq \emptyset$ with probability 1. If $C \cap \alpha \neq \emptyset$, then clearly r_f is accepting with probability 1, and hence is correct with probability 1. Otherwise, $C \subseteq Q_{rej}$, but then we claim that $w_f \in L(\mathcal{A})$ with probability 0. For $i \geq 1$, let w_f^i be the i -th letter of w_f , and for $i \geq 0$ let q_f^i be the i -th state in r_f . Then, by semantically determinism, for all $i \geq 0$, it holds that $w_f \in L(\mathcal{A})$ iff $w_f[i+1, \infty] \in L(\mathcal{A}^{q_f^i})$. Moreover, the word $w_f[i+1, \infty]$ is independent of q_f^i , and hence for all $q \in Q$ and $i \geq 0$, the event $w_f[i+1, \infty] \in L(\mathcal{A}^q)$ is independent of q_f^i . Thus, for all $q \in Q$ and $i \geq 0$, it holds that $\mathbb{P}(w_f \in L(\mathcal{A}) | q_f^i = q) = \mathbb{P}(w_f[i+1, \infty] \in L(\mathcal{A}^q)) = \mathbb{P}(L(\mathcal{A}^q))$. Hence, by definition of Q_{rej} , and by the fact that Q_{rej} is finite, we have that $\mathbb{P}(w_f \in L(\mathcal{A}) | r_f^i \in Q_{rej}) = 0$ for all $i \geq 0$, and so $\mathbb{P}(w_f \in L(\mathcal{A}) | r_f \text{ visits } Q_{rej}) = 0$. Overall, we showed that $\mathbb{P}(r_f \text{ is correct}) = 1$.

Notice that $\mathbb{P}(L(\mathcal{A}) \setminus L(\mathcal{A}^f))$, is precisely the probability that a random word w_f is in $L(\mathcal{A})$ but not accepted by \mathcal{A}^f . Namely, the probability that r_f is not correct. Hence, $\mathbb{P}(L(\mathcal{A}) \setminus L(\mathcal{A}^f)) = 0$, and \mathcal{A} is almost-DBP. ◀

We continue to co-Büchi automata and show that unlike the case of Büchi, here semantic determinism does not imply almost-DBPness.

► **Theorem 11.** *There is an SD-NCW that is not almost-DBP.*

Proof. Consider the NCW \mathcal{A}_2 in Figure 5. It is not hard to see that $L(\mathcal{A}_2) = \{a, b\}^\omega$, and hence $\mathbb{P}(L(\mathcal{A}_2)) = 1$. In fact all the states q of \mathcal{A}_2 have $L(\mathcal{A}_2^q) = \{a, b\}^\omega$, and so it is semantically deterministic. Moreover, every deterministic pruning of \mathcal{A}_2 is strongly connected and not α -free. It follows that any deterministic pruning of \mathcal{A}_2 recognizes a language of measure zero, and hence \mathcal{A}_2 is not almost-DBP. As an example, consider the deterministic pruning \mathcal{A}'_2 described on the right hand side of Figure 5. It is easy to see that \mathcal{A}'_2 is strongly connected and not α -free, and as such, $\mathbb{P}(L(\mathcal{A}'_2)) = 0$. ◀



■ **Figure 5** An SD-NCW that is not almost-DBP.

Consider a language $L \subseteq \Sigma^\omega$ of infinite words. We say that a finite word $x \in \Sigma^*$ is a *good prefix* for L if $x \cdot \Sigma^\omega \subseteq L$. Then, L is a *co-safety* language if every word in L has a good prefix [1]. Let $co\text{-safe}(L) = \{x \cdot w \in \Sigma^\omega : x \text{ is a good prefix of } L\}$. Clearly, $co\text{-safe}(L) \subseteq L$. The other direction is not necessarily true. For example, if $L \subseteq \{a, b\}^\omega$ is the set of all words with infinitely many a 's, then $co\text{-safe}(L) = \emptyset$. In fact, $co\text{-safe}(L) = L$ iff L is a co-safety language. As we show now, when L is NCW-recognizable, we can relate L and $co\text{-safe}(L)$ as follows.

► **Lemma 12.** *If L is NCW-recognizable, then $\mathbb{P}(L(\mathcal{A}) \setminus co\text{-safe}(L(\mathcal{A}))) = 0$.*

Proof. Consider an NCW-recognizable language L . Since $NCW=DCW$, there is a DCW \mathcal{D} that recognizes $L(\mathcal{A})$. Assume without loss of generality that \mathcal{D} has a single state q with $L(\mathcal{A}^q) = \Sigma^\omega$, in particular, $C = \{q\}$ is the only ergodic α -free SCC of \mathcal{A} . Then, for every word $w \in \Sigma^\omega$, we have that $w \in co\text{-safe}(L)$ iff the run of \mathcal{D} on w reaches C . Hence, the probability that $w \in L(\mathcal{A}) \setminus co\text{-safe}(L(\mathcal{A}))$ equals the probability that $inf(r)$ is α -free but is not an ergodic SCC of \mathcal{D} . Since the later happens w.p 0, we have that $\mathbb{P}(L(\mathcal{A}) \setminus co\text{-safe}(L(\mathcal{A}))) = 0$. ◀

By Lemma 12, pruning an NCW in a way that would make it recognize $co\text{-safe}(L(\mathcal{A}))$ results in a DCW that approximates \mathcal{A} , and thus witnesses that \mathcal{A} is almost-DBP. We now show that for GFG-NCWs, such a pruning is possible, and conclude that GFG-NCWs are almost-DBP.

► **Theorem 13.** *All GFG-NCWs are almost-DBP.*

Proof. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ be a GFG-NCW. Consider the NCW $\mathcal{A}' = \langle \Sigma, Q, q_0, \delta, \alpha' \rangle$, where $\alpha' = \alpha \cup \{q \in Q : L(\mathcal{A}^q) \neq \Sigma^\omega\}$. We prove that \mathcal{A}' is a GFG-NCW with $L(\mathcal{A}') = co\text{-safe}(L(\mathcal{A}))$. Consider a word $w \in L(\mathcal{A}')$, and let $r = q_0, q_1, q_2, \dots$ be an accepting run of \mathcal{A}' on w . There exists a prefix $x \in \Sigma^*$ of w such that r reaches some $q \notin \alpha'$ when reading x . Hence $L(\mathcal{A}^q) = \Sigma^\omega$, and so $x \cdot \Sigma^\omega \subseteq L(\mathcal{A})$. That is, x is a good prefix and $w \in co\text{-safe}(L(\mathcal{A}))$. Thus, $L(\mathcal{A}') \subseteq co\text{-safe}(L(\mathcal{A}))$.

In order to see that \mathcal{A}' is GFG and that $co\text{-safe}(L(\mathcal{A})) \subseteq L(\mathcal{A}')$, we consider a GFG strategy f of \mathcal{A} and use it as a strategy for \mathcal{A}' . We need to prove that for all $w \in co\text{-safe}(L(\mathcal{A}))$, the run r that f generates on w eventually visits only states $q \notin \alpha'$. Since $co\text{-safe}(L(\mathcal{A})) \subseteq L(\mathcal{A})$, we know that $inf(r) \cap \alpha = \emptyset$. It is left to show that r eventually visits only states $q \in Q$ with $L(\mathcal{A}^q) = \Sigma^\omega$. Observe that if $x \in \Sigma^*$ is a good prefix of $L(\mathcal{A})$, then for all $y \in \Sigma^*$, we have that $x \cdot y$ is also a good prefix. Moreover, if $x \in \Sigma^*$ is a good prefix, then since f is a GFG strategy, it follows that for all $u \in \Sigma^\omega$ the run $f(x \cdot u)$ is accepting, and hence $u \in L(\mathcal{A}^{f(x)})$. I.e., $L(\mathcal{A}^{f(x)}) = \Sigma^\omega$. Thus, w has only finitely many bad prefixes, and so $f(x) \in \{q \in Q : L(\mathcal{A}^q) \neq \Sigma^\omega\}$ for only finitely many prefixes x of w . That is, $inf(r) \cap \alpha' = \emptyset$, and f is a GFG strategy for \mathcal{A}' .

So, \mathcal{A}' is a GFG-NCW with $L(\mathcal{A}') = \text{co-safe}(L(\mathcal{A}))$. Since $\text{co-safe}(L(\mathcal{A}))$ is co-safe, it is DWW-recognizable [27]. By [7], GFG-NCWs whose language is DWW-realizable are DBP. Let δ' be the restriction δ to a deterministic transition function such that $\mathcal{D}' = \langle \Sigma, Q, q_0, \delta', \alpha' \rangle$ is a DCW with $L(\mathcal{D}') = L(\mathcal{A}') = \text{co-safe}(L(\mathcal{A}))$. Consider now the DCW $\mathcal{D} = \langle \Sigma, Q, q_0, \delta', \alpha \rangle$ that is obtained from \mathcal{D}' by replacing α' with α . It is clear that \mathcal{D} is a pruning of \mathcal{A} . Note that, $\alpha \subseteq \alpha'$, and hence $\text{co-safe}(L(\mathcal{A})) = L(\mathcal{D}') \subseteq L(\mathcal{D})$. That is, \mathcal{D} is a pruning of \mathcal{A} that approximates $L(\mathcal{A})$ up to a negligible set, and \mathcal{A} is almost-DBP. ◀

References

- 1 B. Alpern and F.B. Schneider. Recognizing safety and liveness. *Distributed computing*, 2:117–126, 1987.
- 2 B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.
- 3 G. Avni and O. Kupferman. Stochastization of weighted automata. In *40th Int. Symp. on Mathematical Foundations of Computer Science*, volume 9234 of *Lecture Notes in Computer Science*, pages 89–102. Springer, 2015.
- 4 M. Bagnol and D. Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proc. 38th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 122 of *LIPICs*, pages 16:1–16:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
- 5 U. Boker, D. Kuperberg, O. Kupferman, and M. Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proc. 40th Int. Colloq. on Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 89–100, 2013.
- 6 U. Boker, D. Kuperberg, K. Lehtinen, and M. Skrzypczak. On the succinctness of alternating parity good-for-games automata. In *Proc. 40th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 182 of *LIPICs*, pages 41:1–41:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 7 U. Boker, O. Kupferman, and M. Skrzypczak. How deterministic are Good-For-Games automata? In *Proc. 37th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 93 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 18:1–18:14, 2017.
- 8 U. Boker and K. Lehtinen. Good for games automata: From nondeterminism to alternation. In *Proc. 30th Int. Conf. on Concurrency Theory*, volume 140 of *LIPICs*, pages 19:1–19:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- 9 J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.
- 10 K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Simple stochastic parity games. In *Proc. 12th Annual Conf. of the European Association for Computer Science Logic*, volume 2803, pages 100–113. Springer, 2003.
- 11 Th. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th Int. Colloq. on Automata, Languages, and Programming*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009.
- 12 T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.
- 13 D. Kuperberg and A. Majumdar. Width of non-deterministic automata. In *Proc. 35th Symp. on Theoretical Aspects of Computer Science*, volume 96 of *LIPICs*, pages 47:1–47:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 14 D. Kuperberg and M. Skrzypczak. On determinisation of good-for-games automata. In *Proc. 42nd Int. Colloq. on Automata, Languages, and Programming*, pages 299–310, 2015.

- 15 O. Kupferman. Automata theory and model checking. In *Handbook of Model Checking*, pages 107–151. Springer, 2018.
- 16 O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.
- 17 O. Kupferman and M.Y. Vardi. From linear time to branching time. *ACM Transactions on Computational Logic*, 6(2):273–294, 2005.
- 18 O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.
- 19 L.H. Landweber. Decision problems for ω -automata. *Mathematical Systems Theory*, 3:376–384, 1969.
- 20 K. Lehtinen and M. Zimmermann. Good-for-games ω -pushdown automata. In *Proc. 35th IEEE Symp. on Logic in Computer Science*, 2020.
- 21 S. Miyano and T. Hayashi. Alternating finite automata on ω -words. *Theoretical Computer Science*, 32:321–330, 1984.
- 22 G. Morgenstern. Expressiveness results at the bottom of the ω -regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.
- 23 D.E. Muller, A. Saoudi, and P. E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, pages 422–427, 1988.
- 24 D. Niwinski and I. Walukiewicz. Relating hierarchies of word and tree automata. In *Proc. 15th Symp. on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*. Springer, 1998.
- 25 M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.
- 26 S. Safra. On the complexity of ω -automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.
- 27 A.P. Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6:495–511, 1994.
- 28 A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.
- 29 M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

A Determinization of a SD-NBW

Given an SD-NBW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, the DBW generated in [14] is $\mathcal{A}' = \langle \Sigma, Q', q'_0, \delta', \alpha' \rangle$, where $Q' = 2^Q$, $q'_0 = \{q_0\}$, $\alpha' = \{S \in 2^Q : S \subseteq \alpha\}$, and the transition function δ' is defined for every subset $S \in 2^Q$ and letter $\sigma \in \Sigma$ as follows. If $\delta(S, \sigma) \cap \alpha = \emptyset$, then $\delta'(S, \sigma) = \delta(S, \sigma)$. Otherwise, if $\delta(S, \sigma) \cap \alpha \neq \emptyset$, then $\delta'(S, \sigma) = \delta(S, \sigma) \cap \alpha$.

Thus, we proceed as the standard subset construction, except that whenever a constructed set contains a state in α , we leave in the set only states in α . Accordingly, every reachable state $S \in Q'$ contains only α -states of \mathcal{A} or only $\bar{\alpha}$ -states of \mathcal{A} . Note that as \mathcal{A} is SD, then for every two states $q, q' \in Q$, letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, s \rangle, \langle q', \sigma, s' \rangle \in \Delta$, if $q \sim_{\mathcal{A}} q'$, then $s \sim_{\mathcal{A}} s'$. Consequently, every reachable state S of \mathcal{A}' consists of \mathcal{A} -equivalent states. Without loss of generality, we restrict \mathcal{A}' to its reachable states.

The following two propositions follow immediately from the definitions:

► **Proposition 14.** *Consider states $q \in Q$ and $S \in Q'$, a letter $\sigma \in \Sigma$, and transitions $\langle q, \sigma, q' \rangle$ and $\langle S, \sigma, S' \rangle$ of \mathcal{A} and \mathcal{A}' , respectively. If q is \mathcal{A} -equivalent to the states in S , then q' is \mathcal{A} -equivalent to the states in S' .*

► **Proposition 15.** *Consider a state S of \mathcal{A}' and a letter $\sigma \in \Sigma$. If $\langle S, \sigma, S' \rangle \in \Delta'$ and $S' \notin \alpha'$, then all the σ -successors of a state $s \in S$ are in $S' \setminus \alpha$.*

We can now prove the correctness of the construction:

► **Proposition 16.** *The automata \mathcal{A} and \mathcal{A}' are equivalent.*

Proof. We first prove that $L(\mathcal{A}') \subseteq L(\mathcal{A})$. Let $r_{\mathcal{A}'} = S_0, S_1, S_2, \dots$ be an accepting run of \mathcal{A}' on a word $w = \sigma_1 \cdot \sigma_2 \cdot \dots$. We construct an accepting run of \mathcal{A} on w . Since $r_{\mathcal{A}'}$ is accepting, there are infinitely many positions j_1, j_2, \dots with $S_{j_i} \in \alpha'$. We also define $j_0 = 0$. Consider the DAG $G = \langle V, E \rangle$, where

- $V \subseteq Q \times \mathbb{N}$ is the union $\bigcup_{i \geq 0} (S_{j_i} \times \{i\})$.
- $E \subseteq \bigcup_{i \geq 0} (S_{j_i} \times \{i\}) \times (S_{j_{i+1}} \times \{i+1\})$ is such that for all $i \geq 0$, it holds that $E(\langle s', i \rangle, \langle s, i+1 \rangle)$ iff there is a finite run from s' to s over $w[j_i + 1, j_{i+1}]$. Then, we label this edge by the run from s' to s .

By the definition of \mathcal{A}' , for every $j \geq 0$ and state $s_{j+1} \in S_{j+1}$, there is a state $s_j \in S_j$ such that $\langle s_j, \sigma_j, s_{j+1} \rangle \in \Delta$. Thus, it follows by induction that for every $i \geq 0$ and state $s_{i+1} \in S_{j_{i+1}}$, there is a state $s_i \in S_{j_i}$ such that there is a finite run from s_i to s_{i+1} on $w[j_i + 1, j_{i+1}]$. Thus, the DAG G has infinitely many reachable vertices from the vertex $\langle q_0, 0 \rangle$. Also, as the nondeterminism degree of \mathcal{A} is finite, so is the branching degree of G . Thus, by König's Lemma, G includes an infinite path, and the labels along the edges of this path define a run of \mathcal{A} on w . Since for all $i \geq 1$, the state S_{j_i} is in α' , and so all the states in S_{j_i} are in α , this run is accepting, and we are done.

For the other direction, assume that $w = \sigma_1 \cdot \sigma_2 \cdot \dots \in L(\mathcal{A})$, and let $r = r_0, r_1, \dots$ be an accepting run of \mathcal{A} on w . Let S_0, S_1, S_2, \dots be the run of \mathcal{A}' on w , and assume, by way of contradiction, that there is a position $j \geq 0$ such that S_j, S_{j+1}, \dots is an α -free run on the suffix $w[j + 1, \infty]$. Then, an iterative application of Proposition 15 implies that all the runs of a state $s_j \in S_j$ on $w[j + 1, \infty]$ are α -free in \mathcal{A} . Also, an iterative application of Proposition 14 implies that $r_j \sim_{\mathcal{A}} s_j$, and since r is an accepting run of \mathcal{A} , it holds that \mathcal{A}^{s_j} has an accepting run on $w[j + 1, \infty]$, and we have reached a contradiction. ◀

It is left to prove that weakness of \mathcal{A} is preserved in \mathcal{A}' .

► **Proposition 17.** *If \mathcal{A} is an NWW, then \mathcal{A}' is a DWW.*

Proof. Assume by way of contradiction that there are reachable states $S \in \alpha'$ and $S' \notin \alpha'$, and an infinite run $r_{\mathcal{A}'} = S_0, S_1, S_2, \dots$ that visits both S and S' infinitely often. Recall that a reachable state in Q' contains only α -states of \mathcal{A} or only $\bar{\alpha}$ -states of \mathcal{A} . Hence, S' contains only $\bar{\alpha}$ -states of \mathcal{A} .

As in the proof of Proposition 16, the run $r_{\mathcal{A}'}$ induces an infinite run $r_{\mathcal{A}} = s_0, s_1, s_2, \dots$, where for all positions $j \geq 0$, it holds that $s_j \in S_j$. Since the run $r_{\mathcal{A}'}$ visits S infinitely often, then $r_{\mathcal{A}}$ visits infinitely many α -states. Likewise, since $r_{\mathcal{A}'}$ visits S' infinitely often, then $r_{\mathcal{A}}$ also visits infinitely many $\bar{\alpha}$ -states. This contradicts the weakness of \mathcal{A} , and we are done. ◀

B Details of the Reduction in Theorem 3

We describe the technical details of the construction of \mathcal{A} . Let $T = \langle \Gamma, Q, \rightarrow, q_0, q_{acc}, q_{rej} \rangle$, where Γ is the working alphabet, Q is the set of states, $\rightarrow \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ is the transition relation (we use $(q, a) \rightarrow (q', b, \Delta)$ to indicate that when T is in state q and it reads the input a in the current tape cell, it moves to state q' , writes b in the current tape cell, and its reading head moves one cell to the left/right, according to Δ), q_0 is the initial

state, q_{acc} is the accepting states, and q_{rej} is the rejecting one. The transitions function \rightarrow is defined also for the final states q_{acc} and q_{rej} : when a computation of T reaches them, it erases the tape, goes to the leftmost cell in the tape, and moves to the initial state q_0 . Recall that $s : \mathbb{N} \rightarrow \mathbb{N}$ is the polynomial space function of T . Thus, when T runs on the empty tape, it uses at most $n_0 = s(0)$ cells.

We encode a configuration of T on a word of length at most n_0 by a word of the form $\#\gamma_1\gamma_2 \dots (q, \gamma_i) \dots \gamma_{n_0}$. That is, a configuration starts with $\#$, and all its other letters are in Γ , except for one letter in $Q \times \Gamma$. The meaning of such a configuration is that the j 'th cell in T , for $1 \leq j \leq n_0$, is labeled γ_j , the reading head points at cell i , and T is in state q . For example, the initial configuration of T is $\#(q_0, b)b \dots b$ (with $n_0 - 1$ occurrences of b 's) where b stands for an empty cell. We can now encode a computation of T by a sequence of configurations.

Let $\Sigma = \{\#\} \cup \Gamma \cup (Q \times \Gamma)$ and let $\#\sigma_1 \dots \sigma_{n_0} \#\sigma'_1 \dots \sigma'_{n_0}$ be two successive configurations of T . We also set σ_0, σ'_0 , and σ_{n_0+1} to $\#$. For each triple $\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle$ with $1 \leq i \leq n_0$, we know, by the transition relation of T , what σ'_i should be. In addition, the letter $\#$ should repeat exactly every $n_0 + 1$ letters. Let $next(\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle)$ denote our expectation for σ'_i . That is,

- $next(\langle \gamma_{i-1}, \gamma_i, \gamma_{i+1} \rangle) = next(\langle \#, \gamma_i, \gamma_{i+1} \rangle) = next(\langle \gamma_{i-1}, \gamma_i, \# \rangle) = \gamma_i$.
- $next(\langle (q, \gamma_{i-1}), \gamma_i, \gamma_{i+1} \rangle) = next(\langle (q, \gamma_{i-1}), \gamma_i, \# \rangle) = \begin{cases} \gamma_i & \text{If } (q, \gamma_{i-1}) \rightarrow (q', \gamma'_{i-1}, L) \\ (q', \gamma_i) & \text{If } (q, \gamma_{i-1}) \rightarrow (q', \gamma'_{i-1}, R) \end{cases}$
- $next(\langle \gamma_{i-1}, (q, \gamma_i), \gamma_{i+1} \rangle) = next(\langle \#, (q, \gamma_i), \gamma_{i+1} \rangle) = next(\langle \gamma_{i-1}, (q, \gamma_i), \# \rangle) = \gamma'_i$ where $(q, \gamma_i) \rightarrow (q', \gamma'_i, \Delta)$ ⁸.
- $next(\langle \gamma_{i-1}, \gamma_i, (q, \gamma_{i+1}) \rangle) = next(\langle \#, \gamma_i, (q, \gamma_{i+1}) \rangle) = \begin{cases} \gamma_i & \text{If } (q, \gamma_{i+1}) \rightarrow (q', \gamma'_{i+1}, R) \\ (q', \gamma_i) & \text{If } (q, \gamma_{i+1}) \rightarrow (q', \gamma'_i, L) \end{cases}$
- $next(\langle \sigma_{n_0}, \#, \sigma'_1 \rangle) = \#$.

Consistency with $next$ now gives us a necessary condition for a word to encode a legal computation that uses n_0 tape cells.

In order to accept words that satisfy C1, namely detect a violation of $next$, the NWW \mathcal{A} use its nondeterminism and guesses a triple $\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle \in \Sigma^3$ and guesses a position in the word, where it checks whether the three letters to be read starting this position are σ_{i-1}, σ_i , and σ_{i+1} , and checks whether $next(\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle)$ is not the letter to come $n_0 + 1$ letters later. Once \mathcal{A} sees such a violation, it goes to an accepting sink. If $next$ is respected, or if the guessed triple and position is not successful, then \mathcal{A} returns to its initial state. Also, at any point that \mathcal{A} still waits to guess a position of a triple, it can guess to return back to the initial state.

In order to accept words that satisfy C2, namely detect an encoding of the initial configuration of T on the empty tape and a final configuration after it that is accepting, the NWW \mathcal{A} guesses a position where it compares the next $n_0 + 1$ letters with $\#(q_0, b)b \dots b$. If the initial configuration is indeed detected, it waits for letters in $\{q_{acc}, q_{rej}\} \times \Gamma$. If a letter with q_{acc} arrives before a letter with q_{rej} , then \mathcal{A} goes to the accepting sink. Otherwise if a letter with q_{rej} arrives before a letter with q_{acc} , then \mathcal{A} returns back to the initial state. Also, at any point that \mathcal{A} still waits to detect the initial configuration, or when it waits to

⁸ We assume that the reading head of T does not “fall” from the right or the left boundaries of the tape. Thus, the case where $(i = 1)$ and $(q, \gamma_i) \rightarrow (q', \gamma'_i, L)$ and the dual case where $(i = n_0)$ and $(q, \gamma_i) \rightarrow (q', \gamma'_i, R)$ are not possible.

see a letter in $\{q_{acc}, q_{rej}\} \times \Gamma$, it can guess to return back to the initial state. Note that we could have added the option to keep on waiting for q_{acc} even if q_{rej} arrives first. Indeed, if w includes the initial configuration and both q_{rej} and q_{acc} afterwards, then there must be a violation of *next*.

C Correctness and full details of the reduction in Theorem 4

We first prove that the GFG strategy g defined in Proposition 6 satisfies two essential properties. Then, in Lemma 20, we show that these properties imply that g is a winning GFG strategy for \mathcal{A}_φ .

► **Lemma 18.** *For all $u \in (X \cdot C)^*$ and $v \in R_{n,m}$, if $g(u) = p$, then there is a prefix $y \in (X \cdot C)^*$ of v such that $g(u \cdot y) = q_0$.*

Proof. Let $j \in [m]$ and $2 \leq k \leq n$, be such that v ends with the word $x_k \cdot c_j \cdot x_{k+1} \cdot c_j \cdots x_n \cdot c_j$, and k is the minimal index that is greater than 1, for which x_k participates in c_j . Since we assume that each of the clauses of φ depend on at least two variables, such $k > 1$ exists. Let $i \in \{0, 1\}$ be minimal with $c_j \in C_k^i$, and let $z \in (X \cdot C)^*$ be a prefix of v such that $v = z \cdot x_k \cdot c_j \cdots x_n \cdot c_j$. If there is a prefix $y \in (X \cdot C)^*$ of z , such that $g(u \cdot y) = q_0$ then we are done. Otherwise, $g(u \cdot z) = p$. By definition of g and the choice of k , we know that $g(u \cdot z \cdot x_k) = q_k^i$, where the assignment $x_k = i$ satisfies c_j . Thus, if we take $y = z \cdot x_k \cdot c_j$, then $g(u \cdot y) = q_0$, and y is a prefix of v . ◀

► **Lemma 19.** *For all $u \in (X \cdot C)^*$ and $v \in R_{n,m}$, if $g(u) = q_0$, then there is a prefix $z \in (X \cdot C)^*$ of v such that $g(u \cdot z) = p$.*

Proof. This follows immediately from the fact that $D_{n,m}$ is a DFW that recognizes $R_{n,m}$ and p is the only accepting state of $D_{n,m}$. Thus, we may take z to be the minimal prefix of v that is in $R_{n,m}$. ◀

Recall that a GFG strategy $g : \Sigma^* \rightarrow Q$ has to agree with the the transitions of \mathcal{A}_φ . That is, for all $w \in (X \cdot C)^*$, $x_k \in X$, and $c_j \in C$, it holds that $(g(w), x_k, g(w \cdot x_k))$ and $(g(w \cdot x_k), c_j, g(w \cdot x_k \cdot c_j))$ are in Δ_φ . In addition, if g satisfies the conditions in Lemmas 18 and 19, we say that g supports a (p, q_0) -circle.

► **Lemma 20.** *If $g : \Sigma^* \rightarrow Q$ is consistent with Δ_φ and supports a (p, q_0) -circle, then for all words $w \in L_{n,m}$, the run $g(w)$ is accepting.*

Proof. Consider a word $w \in L_{n,m} = (R_{n,m})^\omega$. Observe that if $w' \in (X \cdot C)^\omega$ is a suffix of w , then $w' \in L_{n,m}$, and hence has a prefix in $R_{n,m}$. Thus, if g supports a (p, q_0) -circle, there exist $y_1, z_1 \in (X \cdot C)^*$, such that $y_1 \cdot z_1$ is a prefix of w , $g(y_1) = q_0$, and $g(y_1 \cdot z_1) = p$. Let $w' \in (X \cdot C)^\omega$ be the suffix of w with $w = y_1 \cdot z_1 \cdot w'$. By the above, $w' \in L_{n,m}$, and we can now apply again the assumption on g to obtain $y_2, z_2 \in (X \cdot C)^*$ such that $y_2 \cdot z_2$ is a prefix of w' , $g(y_1 \cdot z_1 \cdot y_2) = q_0$, and $g(y_1 \cdot z_1 \cdot y_2 \cdot z_2) = p$. By iteratively applying this argument, we construct $\{y_i, z_i : i \geq 1\} \subseteq (X \cdot C)^*$, such that $w^i = y_1 \cdot z_1 \cdot y_2 \cdot z_2 \cdots y_{i-1} \cdot z_{i-1} \cdot y_i$ is a prefix of w , and $g(w^i) = q_0$, for all $i \geq 1$. We conclude that $q_0 \in \inf(g(w))$, and hence $g(w)$ is accepting. ◀

It is easy to see that there is a correspondence between assignments to the variables in X and deterministic prunnings of \mathcal{A}_φ . Indeed, a pruning of p amounts to choosing, for each $k \in [n]$, a value $i_k \in \{0, 1\}$: the assignment $x_k = i_k$ corresponds to keeping the transition $\langle p, x_k, q_k^{i_k} \rangle$ and removing the transition $\langle p, x_k, q_k^{-i_k} \rangle$. For an assignment $a : X \rightarrow \{0, 1\}$, we

denote by \mathcal{A}_φ^a the deterministic pruning of \mathcal{A}_φ that is associated with a . We prove that a satisfies φ iff \mathcal{A}_φ^a is equivalent to \mathcal{A}_φ . Thus, the number of deterministic prunnings of \mathcal{A}_φ that result in a DBW equivalent to \mathcal{A}_φ , equals to the number of assignments that satisfy φ . In particular, φ is satisfiable iff \mathcal{A}_φ is DBP. This concludes the proof of the lower bound in Theorem 4.

► **Proposition 21.** *For every assignment $a : X \rightarrow \{0, 1\}$, we have that $L(\mathcal{A}_\varphi^a) = L(\mathcal{A}_\varphi)$ iff φ is satisfied by a .*

Proof. Assume first that φ is not satisfied by a . We prove that $L_{n,m} \neq L(\mathcal{A}_\varphi^a)$. Let $j \in [m]$ be such that c_j is not satisfied by a . I.e, for all $k \in [n]$ the assignment $x_k = i_k$ does not satisfy c_j . Since q_k^i is reachable in \mathcal{A}_φ^a iff $i = i_k$, and all c_j -labeled transitions from $\{q_k^i : k \in [n]\}$ are to p , it follows that the run of \mathcal{A}_φ^a on $\{x_1 \cdot c_j \cdot x_2 \cdot c_j \cdots x_n \cdot c_j\}^\omega$ never visits q_0 , and hence is rejecting. Thus, $(x_1 \cdot c_j \cdot x_2 \cdot c_j \cdots x_n \cdot c_j)^\omega \in L_{n,m} \setminus L(\mathcal{A}_\varphi^a)$.

For the other direction, we assume that a satisfies φ and prove that $L(\mathcal{A}_\varphi^a) = L_{n,m}$. Let $g^a : \Sigma^* \rightarrow Q$ be the memoryless strategy that correspond to the pruning \mathcal{A}_φ^a . By Lemma 20, it is sufficient proving that g^a supports a (p, q_0) -circle. Note that every strategy for $L_{n,m}$ satisfies Lemma 19. Indeed, the proof only uses the fact that $D_{n,m}$ is a DFW that recognizes $R_{n,m}$ with a single accepting state p . Thus, we only need to prove that g^a satisfies Lemma 18. That is, for all $u \in (X \cdot C)^*$ and $v \in R_{n,m}$, if $g^a(u) = p$, then there is a prefix $y \in (X \cdot C)^*$ of v , such that $g^a(u \cdot y) = q_0$. Consider such words u and v , and let $j \in [m]$ be such that c_j is the last letter of v . Let $k \in [n]$ be the minimal index for which $c_j \in C_k^{i_k}$, and let $z \in (X \cdot C)^*$ be a prefix of v such that $v = z \cdot x_k \cdot c_j \cdot x_{k+1} \cdot c_j \cdots x_n \cdot c_j$. If there exists a prefix $y \in (X \cdot C)^*$ of z such that $g^a(u \cdot y) = q_0$, then we are done. Otherwise, the finite run of \mathcal{A}_φ^a on z from p , returns back to p , and hence $g^a(u \cdot z) = p$. Now $g^a(u \cdot z \cdot x_k) = q_k^{i_k}$, and since $x_k = i_k$ satisfies c_j we have $g^a(u \cdot z \cdot x_k \cdot c_j) = q_0$. Thus, we may take $y = z \cdot x_k \cdot c_j$ which is a prefix of v , and we are done. ◀

Boolean Automata and Atoms of Regular Languages

Hellis Tamm ✉

Department of Software Science, Tallinn University of Technology, Estonia

Abstract

We examine the role that atoms of regular languages play in boolean automata. We observe that the size of a minimal boolean automaton of a regular language is directly related to the number of atoms of the language. We present a method to construct minimal boolean automata, using the atoms of a given regular language. The “illegal” cover problem of the Kameda-Weiner method for NFA minimization implies that using the union operation only to construct an automaton from a cover – as is the case with NFAs –, is not sufficient. We show that by using the union and the intersection operations (without the complementation operation), it is possible to construct boolean automata accepting a given language, for a given maximal cover.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Boolean automaton, Regular language, Atoms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.86

Funding This work was supported by the Estonian Ministry of Education and Research grant IUT33-13 and by the Estonian Research Council grant PRG1210.

Acknowledgements The author is indebted to late Janusz Brzozowski for suggesting the topic and for collaboration during the early stages of this work.

1 Introduction

Nondeterministic finite automata (NFAs) were introduced by Rabin and Scott [19] in 1959, and since then this model of computation has been extended in many ways. Notably, in 1976, Kozen [14] introduced a model of parallel computation based on a generalization of nondeterminism which was later developed into the notion of alternating finite automaton (AFA) by Chandra, Kozen, and Stockmeyer [7]. Independently from this work, Brzozowski and Leiss [5] introduced an equivalent concept of a boolean finite automaton (BFA), using a different notation. Often, the notions of AFA and BFA are used interchangeably. We use the terminology and notation of [5].

The transition function of a BFA uses boolean combinations of its states, generalizing the notion of NFAs which can be interpreted as using unions of states as transition targets. However, the expressive powers of BFAs and NFAs are the same, that is, BFAs only accept regular languages [5, 7]. Importantly, BFAs can succinctly represent regular languages: for any $n \geq 1$ there exists a BFA with n states such that the minimal deterministic finite automaton (DFA) of the same language has $2^{(2^n)}$ states [7, 16]. Also, it is known that any regular language is accepted by an n -state boolean automaton if and only if its reverse language is accepted by a DFA with at most 2^n states [15, 16]. However, there are languages such that their boolean, nondeterministic, and deterministic complexities coincide [17].

Atoms [6] of a regular language L can be considered as its building blocks, since any quotient of L , including L itself, is a disjoint union of atoms. Recently, several old results of automata theory have been revisited using atoms of regular languages: Brzozowski’s double-reversal method for minimizing a DFA [4] was generalized in [6], the Kameda-Weiner



© Hellis Tamm;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 86; pp. 86:1–86:13

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

method of finding a minimal NFA [13] was reinterpreted in terms of atoms and generalized in [21], and lower bound methods for the size of an NFA [3, 10, 11] were presented in terms of quotients and atoms of regular languages in [22].

We examine the role that atoms play in boolean automata. We show that the size of a minimal BFA of a regular language is directly related to the number of atoms of that language. More exactly, we observe that if a language L has m atoms, then a minimal BFA of L has $\lceil \log_2 m \rceil$ states. Furthermore, we show how to construct a minimal BFA using the atoms of a given language. We note that constructions of a minimal BFA presented in the literature [16, 15, 9] have been using a DFA of the reverse language. We think that using atoms to build BFAs is more natural and goes well with the narrative of considering atoms as building blocks of a language. Our method of constructing a minimal BFA using the atoms of a language ensures that the resulting BFA is atomic; that is, the languages associated with the states of a BFA are unions of atoms. Consequently, every regular language has an atomic minimal BFA; however, we also show that a minimal BFA is not necessarily atomic. For comparison, not every language has an atomic minimal NFA [6]. Symmetric difference NFAs – a subclass of boolean automata that use only the symmetric difference operation in the transition function – are known to be able to succinctly represent some regular languages [24]. Interestingly, every minimal symmetric difference NFA is atomic [23].

We revisit the Kameda-Weiner method of NFA minimization [13] which constructs NFAs from grid covers of a special matrix. However, not every cover yields an NFA that would accept a given language. The problem of “illegal” covers of the Kameda-Weiner matrix has been of interest for decades [13, 11, 21, 22]. We show that one can construct a BFA for a given language, using any cover of the Kameda-Weiner matrix. One can see this result as a solution to the problem of interpreting grid covers of the Kameda-Weiner matrix in terms of finite automata accepting a given language. The “illegal” cover problem implies that using the union operation only to construct such an automaton – as is the case with NFAs –, is not sufficient. We show that by using the union and the intersection operations (without the complementation operation), it is possible to construct boolean automata accepting a given language, for a given maximal cover. We note that by a result in [9], for any BFA of n states, there is an equivalent BFA with $2n$ states that uses the union and the intersection operations only. However, in certain cases, our method can produce such a BFA with less states.

We mention that learning regular languages via AFAs has been studied in [1, 2].

We also note that recently, symbolic versions of AFAs and BFAs have been introduced [8, 20], and it has been claimed that in the symbolic setting, these two automata models become importantly different [20].

2 Automata, Languages, and Equations

A *boolean finite automaton (BFA)* is a quintuple $\mathcal{B} = (Q, \Sigma, \delta, f_0, F)$ where $Q = \{q_0, \dots, q_{n-1}\}$ is a finite, non-empty set of *states*, Σ is a finite non-empty *alphabet*, $\delta : Q \times \Sigma \rightarrow B_Q$ is the *transition function*, where B_Q is the free boolean algebra generated by Q , $f_0 \in B_Q$ is the *initial function* in B_Q , and $F \subseteq Q$ is the set of *final states*. We denote the empty word by ε . The transition function is extended to the function $\delta : B_Q \times \Sigma^* \rightarrow B_Q$ as follows. For every $q_i \in Q$, $a \in \Sigma$, $w \in \Sigma^*$, and $f \in B_Q$, $\delta(q_i, \varepsilon) = q_i$, and $\delta(q_i, aw) = f_{i,a}(\delta(q_0, w), \dots, \delta(q_{n-1}, w))$, where $f_{i,a}(q_0, \dots, q_{n-1}) = \delta(q_i, a)$, and $\delta(f, w) = f(\delta(q_0, w), \dots, \delta(q_{n-1}, w))$. Let $\varphi : Q \rightarrow \{0, 1\}$ be defined by setting $\varphi(q_i) = 1$ if $q_i \in F$, and $\varphi(q_i) = 0$ otherwise, for $q_i \in Q$. The *language accepted* by a BFA \mathcal{B} is $L(\mathcal{B}) = \{w \in \Sigma^* \mid \delta(f_0, w)(\varphi(q_0), \dots, \varphi(q_{n-1})) = 1\}$. Two boolean automata are *equivalent* if they accept the same language. The *right language* of a state q of \mathcal{B} is $L(\mathcal{B}_q)$, where $\mathcal{B}_q = (Q, \Sigma, \delta, q, F)$.

If the functions f_0 and $\delta(q, a)$ are unions of states for every $q \in Q$ and $a \in \Sigma$, then \mathcal{B} is a *nondeterministic finite automaton (NFA)*. Traditionally, for NFAs these functions have been presented as sets of states. We prefer to use the traditional notation of an NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$, where Q , Σ , and F are as in BFA, $I \subseteq Q$ is the set of *initial states*, and $\delta : Q \times \Sigma \rightarrow 2^Q$ is the *transition function*. The *reverse* of an NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ is the NFA $\mathcal{N}^R = (Q, \Sigma, \delta^R, F, I)$, where $q \in \delta^R(p, a)$ if and only if $p \in \delta(q, a)$ for $p, q \in Q$ and $a \in \Sigma$.

If $f_0 \in Q$ and $\delta(q, a) \in Q$ for every $q \in Q$ and $a \in \Sigma$, then \mathcal{B} is a *deterministic finite automaton (DFA)*. The *boolean (nondeterministic, deterministic, respectively) complexity* of a regular language L , denoted by $bc(L)$ ($nc(L)$, $dc(L)$, respectively) is the minimal number of states of a boolean (nondeterministic, deterministic, respectively) automaton of L .

A *boolean system of equations (BSE)* B with variables L_0, \dots, L_{n-1} is a set of language equations

$$L_i = \bigcup_{a \in \Sigma} aF_{i,a}(L_0, \dots, L_{n-1}) \cup L_i^\varepsilon, \quad i = 0, \dots, n-1, \quad (1)$$

where $F_{i,a}$ is a boolean function of the variables L_0, \dots, L_{n-1} , $L_i^\varepsilon = \{\varepsilon\}$ if $\varepsilon \in L_i$, and $L_i^\varepsilon = \emptyset$ otherwise, together with the initial function $F_0(L_0, \dots, L_{n-1})$. The language defined by a BSE B is $L(B) = F_0(L_0, \dots, L_{n-1})$.

Any BSE defines a BFA and *vice versa*. There is a one-one correspondence between the state set $Q = \{q_0, \dots, q_{n-1}\}$ of a BFA \mathcal{B} and the set of language variables $\{L_0, \dots, L_{n-1}\}$ of the corresponding BSE B ; there is a transition from $q_i \in Q$ with $a \in \Sigma$ to a boolean function $f_{i,a}$ in BQ if and only if $F_{i,a}$ is the corresponding function of variables $\{L_0, \dots, L_{n-1}\}$ where the *disjunction* (\vee), *conjunction* (\wedge), and *negation* (\neg) operations are replaced by the set operations *union* (\cup), *intersection* (\cap), and *complement* ($\bar{}$), respectively, and the constants 0 and 1 are replaced by \emptyset and Σ^* , respectively, and a similar correspondence is between the initial functions f_0 and F_0 . Also, any state q_i of \mathcal{B} is final if and only if $L_i^\varepsilon = \{\varepsilon\}$. In the rest of the paper, we treat boolean automata and their corresponding systems of equations interchangeably.

The *left quotient*, or simply *quotient*, of a language L by a word $w \in \Sigma^*$ is the language $w^{-1}L = \{x \in \Sigma^* \mid wx \in L\}$. It is well known that the left quotients of L are the right languages of the states of the minimal DFA of L .

An *atom* of a regular language L with quotients K_0, \dots, K_{n-1} is any non-empty language of the form $\widetilde{K}_0 \cap \dots \cap \widetilde{K}_{n-1}$, where \widetilde{K}_i is either K_i or \overline{K}_i , and \overline{K}_i is the complement of K_i with respect to Σ^* [6]. An atom is *initial* if it has L (rather than \overline{L}) as a term; it is *final* if it contains ε . There is exactly one final atom, the atom $\widehat{K}_0 \cap \dots \cap \widehat{K}_{n-1}$, where $\widehat{K}_i = K_i$ if $\varepsilon \in K_i$, and $\widehat{K}_i = \overline{K}_i$ otherwise. If $\overline{K}_0 \cap \dots \cap \overline{K}_{n-1}$ is an atom, then it is called the *negative atom*, all the other atoms are *positive*. Thus atoms of L are pairwise disjoint languages uniquely determined by L ; they define a partition of Σ^* . Every quotient K_i (including L) is a (possibly empty) union of atoms. Hence, atoms can be considered as building blocks of regular languages. We also note that atoms of L are the classes of the *left congruence* \equiv_L of L defined as follows: for $x, y \in \Sigma^*$, $x \equiv_L y$ if for every $u \in \Sigma^*$, $ux \in L$ if and only if $uy \in L$ [12].

Let $A = \{A_0, \dots, A_{m-1}\}$ be the set of atoms of L , let I_A be the set of initial atoms, and let A_{m-1} be the final atom. The *átomaton* of L is the NFA $\mathcal{A} = (A, \Sigma, \alpha, I_A, \{A_{m-1}\})$ where $A_j \in \alpha(A_i, a)$ if and only if $A_j \subseteq a^{-1}A_i$, for all $A_i, A_j \in A$ and $a \in \Sigma$. It was shown in [6] that the atoms of L are the right languages of the states of the átomaton, and that the reverse NFA of the átomaton is the minimal DFA of the reverse language L^R of L .

86:4 Boolean Automata and Atoms of Regular Languages

The BSE corresponding to the átomaton \mathcal{A} , also called the *atom equations*, is as follows:

$$A_i = \bigcup_{a \in \Sigma} a \left(\bigcup_{A_j \subseteq a^{-1}A_i} A_j \right) \cup A_i^\varepsilon, \quad i = 0, \dots, m-1, \quad (2)$$

where $A_i^\varepsilon = \emptyset$ for $i = 0, \dots, m-2$, and $A_{m-1}^\varepsilon = \{\varepsilon\}$, with the initial function $L = \bigcup_{A_i \in I_A} A_i$.
A BFA \mathcal{B} is *atomic* if the right languages of its states are unions of atoms of $L(\mathcal{B})$.

3 Boolean Atoms

Let L be a regular language over Σ and let \mathcal{B} be a BFA of L with variables L_0, \dots, L_{n-1} .

A *boolean atom* of \mathcal{B} is any non-empty language $\widetilde{L}_0 \cap \dots \cap \widetilde{L}_{n-1}$, where \widetilde{L}_i is either L_i or its complement \overline{L}_i with respect to Σ^* . Similarly to the atoms of L , boolean atoms of \mathcal{B} are pairwise disjoint, defining a partition of Σ^* .

We study the relationship between boolean atoms of \mathcal{B} and atoms of L . To avoid confusion between these two notions, we also call the atoms of L the *language atoms*.

► **Proposition 1.** *Every atom of L is a union of boolean atoms of \mathcal{B} .*

Proof. Every quotient of L (including L itself) is obtained as a boolean combination of some L_i 's. Hence, every quotient of L can be expressed as a union of intersections involving uncomplemented and complemented variables from $\{L_0, \dots, L_{n-1}\}$. By adding in the missing variables, every quotient can be expressed as a union of boolean atoms. Also, the complement of any quotient is a union of boolean atoms. Since an intersection of unions of boolean atoms is a union of boolean atoms, the proposition holds. ◀

► **Corollary 2.** *Every boolean atom of \mathcal{B} is a subset of some atom of L .*

Proof. Since both the boolean atoms of \mathcal{B} and the atoms of L define a partition of Σ^* , the corollary follows from Proposition 1. ◀

► **Proposition 3.** *A BFA \mathcal{B} is atomic if and only if its boolean atoms are equal to the atoms of L .*

Proof. First assume that \mathcal{B} is atomic. Then every language L_i is a union of some atoms of L , and so is its complement \overline{L}_i . Therefore, any boolean atom $B_i = \widetilde{L}_0 \cap \dots \cap \widetilde{L}_{n-1}$ is an intersection of unions of language atoms, which is a union of language atoms. On the other hand, by Corollary 2, B_i is a subset of some language atom. We conclude that B_i is equal to some atom of L .

Conversely, if the boolean atoms of \mathcal{B} are equal to the atoms of L , then since every L_i is a union of boolean atoms, it is as well a union of language atoms. Hence, \mathcal{B} is atomic. ◀

We note that boolean atoms are a generalization of *partial atoms* of NFAs, introduced in [6]. More exactly, given an NFA with its language equations, its partial atoms are the boolean atoms of the corresponding BFA.

4 Constructing Minimal Boolean Automata Using Atoms

It is known that if a regular language L is accepted by an n -state BFA, then the reverse language L^R is accepted by a DFA with at most 2^n states [7, 16]. We also recall the following theorem by Leiss [16]:

► **Theorem 4.** *Let \mathcal{D} be a DFA with m states. There exists a BFA with $\lceil \log_2 m \rceil$ states which accepts the reverse language of \mathcal{D} .*

Consequently, a minimal BFA of a language L has $\lceil \log_2 m \rceil$ states, where m is the number of states of the minimal DFA of L^R . Since the átomon of L is isomorphic to the reverse automaton of the minimal DFA of L^R [6], we make the following observation:

► **Theorem 5.** *A minimal BFA of a regular language L has $\lceil \log_2 m \rceil$ states, where m is the number of atoms of L .*

We note that Leiss [16] also describes a method to construct a BFA of L^R with $\lceil \log_2 m \rceil$ states by using a DFA of L with m states. Also, Kozen [15] (p. 327) discusses how to construct an AFA of L with k states by using a DFA of L^R with 2^k states, and vice versa.

We present a method to construct a minimal BFA of a regular language, by using its atoms.

Let L be a regular language over an alphabet Σ , and let $A = \{A_0, \dots, A_{m-1}\}$ be the set of atoms of L , with a subset $I_A \subseteq A$ of initial atoms and the final atom A_{m-1} . Let $k = \lceil \log_2 m \rceil$. We show how to construct an atomic BFA with k variables L_0, \dots, L_{k-1} denoting some (not yet identified) languages over Σ . Let us consider the set S of all intersections in the form $\widetilde{L}_0 \cap \dots \cap \widetilde{L}_{k-1}$ where \widetilde{L}_i is either L_i or \overline{L}_i . Clearly, there are 2^k such intersections, and the union of all these intersections is Σ^* . Also, we note that since $k = \lceil \log_2 m \rceil$, the inequality $m \leq 2^k$ holds.

Let us denote any intersection in S by $X_P = \bigcap_{i \in P} L_i \cap \bigcap_{i \in \overline{P}} \overline{L}_i$, where $P \subseteq \{0, \dots, k-1\}$ and $\overline{P} = \{0, \dots, k-1\} \setminus P$. Now, let us choose any subset $S_m = \{X_{P_0}, \dots, X_{P_{m-1}}\}$ of S consisting of m intersections, and set any $X_{P_j} \in S_m$ to be equal to some atom A_j of L . We note that S_m is the set of boolean atoms of the BFA we will be constructing, and by Proposition 3, this BFA will be atomic. For instance, we may choose $P_0 = \{0, \dots, k-1\}$, $P_1 = \{0, \dots, k-2\}$, $P_2 = \{0, \dots, k-3, k-1\}$, $P_3 = \{0, \dots, k-3\}$, etc., and form the following equations between the boolean atoms and the atoms of L :

$$\begin{aligned} L_0 \cap L_1 \cap \dots \cap L_{k-2} \cap L_{k-1} &= A_0, \\ L_0 \cap L_1 \cap \dots \cap L_{k-2} \cap \overline{L}_{k-1} &= A_1, \\ L_0 \cap L_1 \cap \dots \cap \overline{L}_{k-2} \cap L_{k-1} &= A_2, \\ L_0 \cap L_1 \cap \dots \cap \overline{L}_{k-2} \cap \overline{L}_{k-1} &= A_3, \\ &\dots \\ \widetilde{L}_0 \cap \widetilde{L}_1 \cap \dots \cap \widetilde{L}_{k-2} \cap \widetilde{L}_{k-1} &= A_{m-1}, \end{aligned}$$

where \widetilde{L}_i is L_i if $n_i = 0$, and \widetilde{L}_i is \overline{L}_i if $n_i = 1$, where $n_0 n_1 \dots n_{k-1}$ is the binary representation of the number $m-1$ using k bits. For every $X_{P_j} \notin S_m$, that is, for $j = m, \dots, 2^k - 1$, we let $X_{P_j} = \emptyset$.

Clearly, every language L_i , where $i = 0, \dots, k-1$, is the union of those boolean atoms X_{P_j} , where L_i is uncomplemented, that is, $L_i = \bigcup_{i \in P_j} X_{P_j}$.

We derive boolean equations for L_0, \dots, L_{k-1} , using the equations above together with the atom equations (2):

$$\begin{aligned} L_i &= \bigcup_{i \in P_j} X_{P_j} = \bigcup_{i \in P_j} A_j = \bigcup_{i \in P_j} \left(\bigcup_{a \in \Sigma} a \left(\bigcup_{A_h \subseteq a^{-1} A_j} A_h \right) \cup A_j^\varepsilon \right) = \\ &\quad \bigcup_{a \in \Sigma} a \left(\bigcup_{i \in P_j} \bigcup_{A_h \subseteq a^{-1} A_j} X_{P_h} \right) \cup \bigcup_{i \in P_j} A_j^\varepsilon, \end{aligned}$$

that is, we obtain the equations

$$L_i = \bigcup_{a \in \Sigma} a \left(\bigcup_{i \in P_j} \bigcup_{A_h \subseteq a^{-1}A_j} X_{P_h} \right) \cup L_i^\varepsilon, \quad i = 0, \dots, k-1, \quad (3)$$

where $L_i^\varepsilon = \{\varepsilon\}$ if $i \in P_{m-1}$, and $L_i^\varepsilon = \emptyset$ otherwise.

We also notice that $L = \bigcup_{A_h \subseteq L} A_h = \bigcup_{A_h \subseteq L} X_{P_h}$. That is, since L is a union of some of its atoms A_h , it is the union of the corresponding boolean atoms X_{P_h} . Hence, the equations (3) together with the initial function $L = \bigcup_{A_h \subseteq L} X_{P_h}$ form an atomic minimal BFA of L .

In case our goal would be to obtain an atomic minimal BFA with the initial function $L = L_0$, we would have to be able to assign all the initial atoms of L to boolean atoms which have L_0 (rather than $\overline{L_0}$) as a term, and all the other language atoms to boolean atoms with $\overline{L_0}$. It is not difficult to see that this is possible if and only if the condition

$$m - 2^{\lceil \log_2 m \rceil - 1} \leq |I_A| \leq 2^{\lceil \log_2 m \rceil - 1} \quad (4)$$

holds.

The method described above constructs an atomic minimal BFA of a language. However, there may exist non-atomic minimal BFAs as well. The following example illustrates the constructions of minimal BFAs and presents a case of a non-atomic minimal BFA.

► **Example 6.** Let us consider a regular language L from [6, 18], defined by the following nondeterministic system of equations, with $L = L_0$:

$$\begin{aligned} L_0 &= aL_1 \cup b(L_1 \cup L_2), \\ L_1 &= aL_3 \cup b(L_0 \cup L_3), \\ L_2 &= a(L_0 \cup L_2 \cup L_3) \cup \varepsilon, \\ L_3 &= aL_3 \cup bL_1. \end{aligned}$$

It was shown in [6] that the corresponding NFA is a minimal NFA of L , however, it is not atomic. The language L has 6 atoms, denoted by A, B, C, D, E , and F , from which B, D , and F are the initial atoms, and A is the final atom. The atom equations are as follows:

$$\begin{aligned} A &= a(A \cup B) \cup \varepsilon, \\ B &= aC \cup bA, \\ C &= b(B \cup D), \\ D &= bC, \\ E &= aD, \\ F &= a(E \cup F) \cup b(E \cup F), \end{aligned}$$

with the initial function $L = B \cup D \cup F$. One can verify that $L_0 = B \cup D \cup F$, $L_1 = C \cup E \cup F$, and $L_3 = D \cup E \cup F$, whereas $L_2 = A \cup E \cup F'$, where $F' = a(E \cup F)$ is a subset of F .

The boolean atoms of the original system of equations are found as the following non-empty intersections:

$$\begin{aligned} L_0 \cap L_1 \cap L_2 \cap L_3 &= F', \\ L_0 \cap L_1 \cap \overline{L_2} \cap L_3 &= F \setminus F', \\ L_0 \cap \overline{L_1} \cap \overline{L_2} \cap L_3 &= D, \\ L_0 \cap \overline{L_1} \cap \overline{L_2} \cap \overline{L_3} &= B, \\ \overline{L_0} \cap L_1 \cap L_2 \cap L_3 &= E, \\ \overline{L_0} \cap L_1 \cap \overline{L_2} \cap \overline{L_3} &= C, \\ \overline{L_0} \cap \overline{L_1} \cap L_2 \cap \overline{L_3} &= A. \end{aligned}$$

Hence, the boolean atoms are A, B, C, D, E, F' , and $F \setminus F'$. We note that F is a union of two boolean atoms, while the other language atoms coincide with boolean atoms.

In the following, we drop the union symbols when representing unions of atoms; for example, $B \cup D \cup F$ is denoted by BDF . A minimal BFA of this language has $\lceil \log_2 6 \rceil = 3$ states. We construct an atomic minimal BFA by the approach described above. Let us denote the states of a minimal BFA by variables L_0, L_1, L_2 , and form the following equations:

$$\begin{aligned} L_0 \cap L_1 \cap L_2 &= A, \\ L_0 \cap L_1 \cap \overline{L_2} &= B, \\ L_0 \cap \overline{L_1} \cap L_2 &= C, \\ L_0 \cap \overline{L_1} \cap \overline{L_2} &= D, \\ \overline{L_0} \cap L_1 \cap L_2 &= E, \\ \overline{L_0} \cap L_1 \cap \overline{L_2} &= F, \\ \overline{L_0} \cap \overline{L_1} \cap L_2 &= \emptyset, \\ \overline{L_0} \cap \overline{L_1} \cap \overline{L_2} &= \emptyset. \end{aligned}$$

From these equations we obtain $L_0 = ABCD$, $L_1 = ABEF$, and $L_2 = ACE$. All the variables correspond to final states because the corresponding languages contain the final atom A . Using the atom equations, we get the following equations:

$$\begin{aligned} ABCD &= aABC \cup bABCD \cup \varepsilon, \\ ABEF &= aABCDEF \cup bAEF \cup \varepsilon, \\ ACE &= aABD \cup bBD \cup \varepsilon, \end{aligned}$$

with $L = BDF$. By replacing atoms with the corresponding boolean expressions over the variables L_0, L_1 , and L_2 , and after a few straightforward simplifications, we obtain the following system of equations, with $L = \overline{L_2}$:

$$\begin{aligned} L_0 &= a(L_0 \cap (L_1 \cup L_2)) \cup bL_0 \cup \varepsilon, \\ L_1 &= a\Sigma^* \cup b((L_1 \cap L_2) \cup \overline{L_0}) \cup \varepsilon, \\ L_2 &= a((L_0 \cap L_1) \cup (\overline{L_1} \cap \overline{L_2})) \cup b(L_0 \cap \overline{L_2}) \cup \varepsilon. \end{aligned}$$

These equations form an atomic minimal BFA for L .

We may also obtain a minimal BFA for L with $L = L_0$, since the condition (4) holds for L . For instance, we could have the atom assignments as follows (with the rest of the intersections set to be empty):

$$\begin{aligned} L_0 \cap L_1 \cap L_2 &= B, \\ L_0 \cap L_1 \cap \overline{L_2} &= D, \\ L_0 \cap \overline{L_1} \cap L_2 &= F, \\ \overline{L_0} \cap L_1 \cap L_2 &= A, \\ \overline{L_0} \cap L_1 \cap \overline{L_2} &= C, \\ \overline{L_0} \cap \overline{L_1} \cap L_2 &= E. \end{aligned}$$

Now we obtain the languages $L_0 = BDF$, $L_1 = ABCD$, and $L_2 = ABEF$, where L_1 and L_2 correspond to the final states. Using the atom equations, we get the following equations:

$$\begin{aligned} BDF &= aCEF \cup bACEF, \\ ABCD &= aABC \cup bABCD \cup \varepsilon, \\ ABEF &= aABCDEF \cup bAEF \cup \varepsilon, \end{aligned}$$

with $L = BDF$. Similarly as above, we obtain the following language equations, with $L = L_0$:

$$\begin{aligned} L_0 &= a((\overline{L_0} \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)) \cup b(\overline{L_0} \cup (L_0 \cap \overline{L_1})), \\ L_1 &= a((L_1 \cap L_2) \cup (\overline{L_0} \cap \overline{L_2})) \cup bL_1 \cup \varepsilon, \\ L_2 &= a\Sigma^* \cup b((\overline{L_0} \cap L_2) \cup (L_0 \cap \overline{L_1})) \cup \varepsilon. \end{aligned}$$

These equations form another atomic minimal BFA of L .

We can also construct a non-atomic minimal BFA for L . Let us take $L_0 = BDF$, $L_1 = CEF$, and $L_2 = ADEF'$, where L_2 is a final state. We note that L_2 is not atomic. Using the atom equations and the equation for F' , we get the following equations:

$$\begin{aligned} BDF &= aCEF \cup bACEF, \\ CEF &= aDEF \cup bBDEF, \\ ADEF' &= aABDEF \cup bC \cup \varepsilon, \end{aligned}$$

with $L = BDF$. We obtain the following language equations, with $L = L_0$:

$$\begin{aligned} L_0 &= aL_1 \cup b((\overline{L_0} \cap \overline{L_1} \cap L_2) \cup L_1), \\ L_1 &= a((L_0 \cap L_1) \cup (L_0 \cap L_2) \cup (L_1 \cap L_2)) \cup b(L_0 \cup (L_1 \cap L_2)), \\ L_2 &= a(L_0 \cup L_2) \cup b(\overline{L_0} \cap L_1 \cap \overline{L_2}) \cup \varepsilon. \end{aligned}$$

These equations form a non-atomic minimal BFA of L . ┘

5 Boolean Automata and Quotient-Atom Matrix

Kameda and Weiner [13] studied the problem of finding a minimal NFA of a language L , using a matrix based on the states of the minimal DFAs of L and L^R . They suggested a method of constructing NFAs, utilizing grid covers of this matrix. However, an NFA formed this way does not necessarily accept L . Therefore, the method of finding a minimal NFA tests grid covers of the matrix in the order of increasing size to see if they are “legal”. The first legal NFA, that is, an NFA found to accept L , is a minimal one.

In this section, we revisit the Kameda-Weiner method and its recent reinterpretation in terms of atoms of the language [21]. We will show that if one aims to form a BFA rather than an NFA, then the problem of “illegal” covers disappears. That is, one can form a BFA for L , using any cover of the quotient-atom matrix.

First, we describe the Kameda-Weiner method in terms of quotients and atoms as presented in [21]. It was shown in [21] that the matrix used by Kameda and Weiner can be viewed as the *quotient-atom matrix* of the language, that is, the matrix with its rows corresponding to the non-empty quotients, and the columns, to the positive atoms of the language. Any (i, j) -entry of this matrix is 1 if the quotient K_i has the atom A_j as its subset, and 0 otherwise. A *grid* of the matrix is the direct product $g = P \times R$ of a set P of quotients with a set R of atoms, such that every atom in R is a subset of every quotient in P . A grid $g = P \times R$ is *maximal* if there is no other grid $g' = P' \times R'$ such that $P \subseteq P'$ and $R \subseteq R'$. A *grid cover* of the matrix is a set $G = \{g_0, \dots, g_{k-1}\}$ of grids, such that every 1-entry (K_i, A_j) belongs to some grid g_h of G . A grid cover is maximal if it consists of maximal grids. Clearly, any grid cover can be made maximal by replacing every non-maximal grid $g = P \times R$ in it by the maximal grid $g' = P' \times R'$ such that $P \subseteq P'$ and $R \subseteq R'$.

Let f_G be the function that assigns to every non-empty quotient K_i the subset of G , consisting of grids $g = P \times R$ such that $K_i \in P$. The Kameda-Weiner method constructs the NFA $\mathcal{N}_G = (G, \Sigma, \eta_G, I_G, F_G)$, where G is a maximal grid cover, $I_G = f_G(K_0)$, $g \in F_G$ if and only if $g \in f_G(K_i)$ implies that $\varepsilon \in K_i$, and transition function is computed by the so-called *intersection rule* $\eta_G(g, a) = \bigcap_{K_i \in P} f_G(a^{-1}K_i)$, for a grid $g = P \times R$ in G and $a \in \Sigma$. The NFA \mathcal{N}_G may or may not accept L . A cover G is called “legal” if $L(\mathcal{N}_G) = L$. To find a minimal NFA of a language L , covers of the matrix are tested in the order of increasing size to see if they are legal; the first legal NFA is a minimal one.

In [21], the Kameda-Weiner method was interpreted in terms of atoms, with the key observation being that any maximal grid can be seen as the set of atoms it involves. We call a set $\{L_0, \dots, L_{k-1}\}$ of languages a *language cover* for L if for every L_i there is a quotient

K_j such that $L_i \subseteq K_j$ and if every quotient of L is a union of some L_i 's. A language cover $\{L_0, \dots, L_{k-1}\}$ is *atomic* if every L_i , where $i = 0, \dots, k-1$, is a union of atoms of L . Now, for any grid cover $G = \{g_0, \dots, g_{k-1}\}$ there is a corresponding atomic language cover $C = \{U(R_0), \dots, U(R_{k-1})\}$, where $g_i = P_i \times R_i$ and $U(R_i) = \bigcup_{A_j \in R_i} A_j$ for $i = 0, \dots, k-1$. The following theorem was proved in [21]:

► **Theorem 7.** *Let $G = \{g_0, \dots, g_{k-1}\}$ be a cover consisting of maximal grids $g_i = P_i \times R_i$, $i = 0, \dots, k-1$, and let $\mathcal{N}_G = (G, \Sigma, \eta_G, I_G, F_G)$ be the corresponding NFA, obtained by the intersection method. It holds that $g_i \in I_G$ if and only if $U(R_i) \subseteq L$, and $g_i \in F_G$ if and only if $\varepsilon \in U(R_i)$. For any $g_i, g_j \in G$ and $a \in \Sigma$, $g_j \in \eta_G(g_i, a)$ if and only if the inclusion $U(R_j) \subseteq a^{-1}U(R_i)$ holds.*

By Theorem 7, it is easy to see that forming the NFA \mathcal{N}_G corresponds to defining this NFA with variables L_0, \dots, L_{k-1} using the language equations

$$L_i = \bigcup_{a \in \Sigma} a \left(\bigcup_{U(R_j) \subseteq a^{-1}U(R_i)} L_j \right) \cup L_i^\varepsilon, \quad i = 0, \dots, k-1, \quad (5)$$

where $L_i^\varepsilon = \{\varepsilon\}$ if $\varepsilon \in U(R_i)$, and $L_i^\varepsilon = \emptyset$ otherwise, and the initial function $\bigcup_{U(R_i) \subseteq L} L_i$. We recall that this NFA not necessarily accepts L .

Gruber and Holzer [11] defined a canonical bipartite graph G_L of a language, which is a graph representation of the Kameda-Weiner matrix. By denoting the bipartite dimension of G_L , that is, the minimum number of bicliques to cover the edges of G_L , by $d(G_L)$, it is clear that the size of the minimal grid cover of the quotient-atom matrix is equal to $d(G_L)$. It has been a long-standing problem to identify languages with a legal minimal cover [11, 13, 21, 22].

It is known that $dc(L) \leq 2^{d(G_L)}$ [11]. Since $dc(L^R) = m$, where m is the number of atoms of L [6], and $d(G_{L^R}) = d(G_L)$ [11, 13], we get that $m \leq 2^{d(G_L)}$. By Theorem 5, a minimal BFA of L has $bc(L) = \lceil \log_2 m \rceil$ states, implying that $bc(L) \leq d(G_L)$. Hence, the following statement holds:

► **Theorem 8.** *A minimal BFA of a regular language L has at most $d(G_L)$ states.*

Theorem 8 provides an upper bound to the size of a minimal BFA of a language in terms of the size of a minimal cover of the quotient-atom matrix/graph of the language. Since a minimal NFA has at least $d(G_L)$ states [11, 13, 21], the inequalities

$$bc(L) \leq d(G_L) \leq nc(L)$$

hold. Also, we can state the following corollary:

► **Corollary 9.** *A minimal NFA is a minimal BFA of L if and only if the equalities $bc(L) = d(G_L) = nc(L)$ hold.*

In the following sections, we present two methods to construct a BFA, using a cover of the quotient-atom matrix/graph.

5.1 Constructing a General BFA

Given a language cover $C = \{L_0, \dots, L_{k-1}\}$ of L , we call any non-empty intersection $\widetilde{L}_0 \cap \dots \cap \widetilde{L}_{k-1}$ where \widetilde{L}_i is either L_i or \overline{L}_i , a *cover atom* of C . Similarly to the atoms of L , cover atoms are pairwise disjoint and their union is Σ^* .

The following proposition is a slightly modified version of the result in [6]¹:

¹ The result in [6] concerns the case where the language cover consists of the right languages of an NFA accepting L ; however, its proof holds for any cover.

86:10 Boolean Automata and Atoms of Regular Languages

► **Proposition 10.** *Given a language cover C of L , every cover atom of C is a subset of some atom of L .*

We note that cover atoms of the cover $K = \{K_0, \dots, K_{n-1}\}$ consisting of the quotients of L , are clearly the atoms of L . Moreover, the following proposition holds:

► **Proposition 11.** *The cover atoms of a cover C of L are equal to the atoms of L if and only if C is atomic.*

Proof. Let $C = \{L_0, \dots, L_{k-1}\}$ be an atomic language cover of L , meaning that every $L_i \in C$ is a union of atoms of L . In this case every complement language $\overline{L_i}$, for $i = 0, \dots, k-1$, is also a union of atoms. Therefore, any cover atom as an intersection of unions of atoms, is a union of atoms. However, since by Proposition 10, any cover atom is a subset of some atom of L , we conclude that cover atoms are equal to the atoms of L .

If the cover C is not atomic, then there is some $L_i \in C$ that is not a union of atoms of L . However, since L_i is the union of those cover atoms that have L_i uncomplemented in their intersection, there exists at least one non-atomic cover atom of C . ◀

Let us now consider any grid cover $G = \{g_0, \dots, g_{k-1}\}$ of the quotient-atom matrix and the corresponding language cover $C = \{U(R_0), \dots, U(R_{k-1})\}$ of L , with $g_i = P_i \times R_i$ and $U(R_i) = \bigcup_{A_j \in R_i} A_j$ for $i = 0, \dots, k-1$. Let the set of cover atoms of C be $X = \{X_0, \dots, X_{\ell-1}\}$. Since the cover C is atomic, by Proposition 11, the cover atoms of C are the atoms of L . Hence, any atom A_h of L can be expressed as $A_h = \bigcap_{A_h \in R_i} U(R_i) \cap \bigcap_{A_h \notin R_i} \overline{U(R_i)}$.

Now, we can form a BFA with the set $\{L_0, \dots, L_{k-1}\}$ of variables, using the correspondence between the variables L_i and the languages $U(R_i)$, where $i = 0, \dots, k-1$. We obtain the following equations for a BFA, using the atom expressions above and the atom equations (2):

$$L_i = \bigcup_{a \in \Sigma} a \left(\bigcup_{A_j \in R_i} \bigcup_{A_h \subseteq a^{-1}A_j} \left(\bigcap_{A_h \in R_i} L_i \cap \bigcap_{A_h \notin R_i} \overline{L_i} \right) \right) \cup L_i^\varepsilon, \quad i = 0, \dots, k-1, \quad (6)$$

where $L_i^\varepsilon = \{\varepsilon\}$ if $\varepsilon \in U(R_i)$, and $L_i^\varepsilon = \emptyset$ otherwise. The initial function of the BFA is $L = \bigcup_{U(R_i) \subseteq L} L_i$. Clearly, this BFA is atomic, because every L_i is a union of atoms. We also note that boolean atoms of this BFA are equal to the cover atoms.

5.2 Constructing a BFA Without Complementation

We show that any maximal grid cover can be used to construct a BFA that uses the union and the intersection operations only, without a need for the complementation operation. We use the following definition from [21]:

► **Definition 12.** *A set R of atoms is maximal if $R = \{A_j \mid A_j \subseteq \bigcap_{U(R) \subseteq K_i} K_i\}$, where $U(R) = \bigcup_{A_j \in R} A_j$ is the union of atoms in R .*

We observe that the intersection of any quotients of L corresponds to a maximal set of atoms:

► **Proposition 13.** *Given any set $P \subseteq K$ of quotients of L , the set $R = \{A_j \mid A_j \subseteq \bigcap_{K_h \in P} K_h\}$ of atoms in their intersection, is maximal.*

Proof. Let $P \subseteq K$ be any set of quotients of L and let $R = \{A_j \mid A_j \subseteq \bigcap_{K_h \in P} K_h\}$ be the set of atoms in their intersection. Since $U(R) = \bigcap_{K_h \in P} K_h = \bigcap_{U(R) \subseteq K_i} K_i$, we get that $R = \{A_j \mid A_j \subseteq \bigcap_{U(R) \subseteq K_i} K_i\}$. Thus, R is maximal. ◀

► **Proposition 14.** *If a set R of atoms is maximal, then the set $\{A_j \mid A_j \subseteq a^{-1}U(R)\}$, where $a \in \Sigma$, is also maximal.*

Proof. Let R be a maximal set of atoms and $a \in \Sigma$. Then $U(R) = \bigcap_{U(R) \subseteq K_i} K_i$ and we get $a^{-1}U(R) = a^{-1} \bigcap_{U(R) \subseteq K_i} K_i = \bigcap_{U(R) \subseteq K_i} a^{-1}K_i$. Since for any quotient K_i of L , $a^{-1}K_i$ is also a quotient of L , the language $a^{-1}U(R)$ is equal to an intersection of some quotients. By Proposition 13, we conclude that the set $\{A_j \mid A_j \subseteq a^{-1}U(R)\}$ is maximal. ◀

Let us consider a maximal grid cover $G = \{g_0, \dots, g_{k-1}\}$ of the quotient-atom matrix and the corresponding language cover $C = \{U(R_0), \dots, U(R_{k-1})\}$ of L , with $g_i = P_i \times R_i$ and $U(R_i) = \bigcup_{A_j \in R_i} A_j$ for $i = 0, \dots, k-1$. It is known that any maximal grid involves a maximal set of atoms [21]. Hence, every R_i , where $i = 0, \dots, k-1$, is a maximal set of atoms. We denote by $P_{i,a}$, where $a \in \Sigma$, the set $P_{i,a} = \{K_h \mid a^{-1}U(R_i) \subseteq K_h\}$ of those quotients of L that include $a^{-1}U(R_i)$ as a subset. By Proposition 14, the equality $a^{-1}U(R_i) = \bigcap_{K_h \in P_{i,a}} K_h$ holds. Also, we recall that every quotient K_h is a union of some elements of the cover C , that is, $K_h = \bigcup_{U(R_i) \subseteq K_h} U(R_i)$.

We construct a BFA with the set $\{L_0, \dots, L_{k-1}\}$ of variables, using the correspondence between the variables L_i and the languages $U(R_i)$, where $i = 0, \dots, k-1$. The language equations are as follows:

$$L_i = \bigcup_{a \in \Sigma} a \left(\bigcap_{K_h \in P_{i,a}} \bigcup_{U(R_j) \subseteq K_h} L_j \right) \cup L_i^\varepsilon, \quad i = 0, \dots, k-1, \quad (7)$$

where $L_i^\varepsilon = \{\varepsilon\}$ if $\varepsilon \in U(R_i)$, and $L_i^\varepsilon = \emptyset$ otherwise. The initial function is $L = \bigcup_{U(R_i) \subseteq L} L_i$. This BFA is atomic, since every L_i is a union of atoms of L .

We note that the BFA corresponding to the equations (7) uses only the union and the intersection operations. That is, the complementation operation is not needed when we use a maximal cover as a basis for the states of a BFA.

One can see this result as a solution to the problem of interpreting grid covers of the Kameda-Weiner matrix, or equivalently, biclique edge covers of the quotient-atom graph, in terms of finite automata accepting a given language. The “illegal” cover problem mentioned above implies that using the union operation only to construct such an automaton – as is the case with NFAs –, is not sufficient. However, we showed that with the union and the intersection operations it is possible to construct boolean automata accepting a given language, for a given maximal cover.

We note that by a result in [9], for any BFA of n states, there is an equivalent BFA with $2n$ states that uses the union and the intersection operations only. However, since the inequality $bc(L) \leq d(G_L)$ holds for any language L , and because we can construct a BFA with $d(G_L)$ states, using the union and the intersection operations only, our method can produce such a BFA for any language L for which the inequality $d(G_L) < 2bc(L)$ holds, with less states.

6 Conclusions

We have started a study of the role that atoms of a regular language have in the context of boolean automata, their relationship with boolean atoms, and how they can be used to construct BFAs.

The problem of “illegal” covers of the Kameda-Weiner matrix used for NFA minimization has been of interest for a long time. We presented a new interpretation of the covers in terms of BFAs, so that every cover becomes “legal”. We showed that it is sufficient to use the union and the intersection operations to construct a BFA for a given language, corresponding to a maximal cover of the matrix. Moreover, the resulting BFAs are atomic.

These are part of the evidence of the significant role that atoms play in the theory of automata.

References

- 1 Dana Angluin, Sarah Eisenstat, and Dana Fisman. Learning regular languages via alternating automata. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25–31, 2015*, pages 3308–3314. AAAI Press, 2015.
- 2 Sebastian Berndt, Maciej Liskiewicz, Matthias Lutter, and Rüdiger Reischuk. Learning residual alternating automata. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA*, pages 1749–1755. AAAI Press, 2017.
- 3 Jean-Camille Birget. Intersection and union of regular languages and state complexity. *Inf. Process. Lett.*, 43(4):185–190, 1992.
- 4 Janusz A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In *Proc. Symp. on Mathematical Theory of Automata*, volume 12 of *MRI Symposia Series*, pages 529–561. Polytechnic Press, Polytechnic Institute of Brooklyn, N.Y., 1963.
- 5 Janusz A. Brzozowski and Ernst L. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theor. Comput. Sci.*, 10:19–35, 1980.
- 6 Janusz A. Brzozowski and Hellis Tamm. Theory of átomata. *Theor. Comput. Sci.*, 539:13–27, 2014.
- 7 Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.
- 8 Loris D’Antoni, Zachary Kincaid, and Fang Wang. A symbolic decision procedure for symbolic alternating finite automata. In *Proceedings of the Thirty-Fourth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2018, Dalhousie University, Halifax, Canada, June 6–9, 2018*, volume 341 of *Electronic Notes in Theoretical Computer Science*, pages 79–99. Elsevier, 2018.
- 9 Abdelaziz Fellah, Helmut Jürgensen, and Sheng Yu. Constructions for alternating finite automata. *Int. J. of Comput. Math.*, 35(1-4):117–132, 1990.
- 10 Ian Glaister and Jeffrey O. Shallit. A lower bound technique for the size of nondeterministic finite automata. *Inf. Process. Lett.*, 59(2):75–77, 1996.
- 11 Hermann Gruber and Markus Holzer. Finding lower bounds for nondeterministic state complexity is hard. In *Proc. of DLT 2006*, volume 4036 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2006.
- 12 Szabolcs Iván. Complexity of atoms, combinatorially. *Inf. Process. Lett.*, 116(5):356–360, 2016.
- 13 Tsunehiko Kameda and Peter Weiner. On the state minimization of nondeterministic finite automata. *IEEE Trans. Computers*, 19(7):617–627, 1970.
- 14 Dexter Kozen. On parallelism in Turing machines. In *17th Annual Symposium on Foundations of Computer Science, Houston, Texas, USA, 25–27 October 1976*, pages 89–97, 1976.
- 15 Dexter Kozen. *Theory of Computation*. Texts in Computer Science. Springer, 2006.
- 16 Ernst L. Leiss. Succinct representation of regular languages by boolean automata. *Theor. Comput. Sci.*, 13:323–330, 1981.
- 17 Ernst L. Leiss. Succinct representation of regular languages by boolean automata II. *Theor. Comput. Sci.*, 38:133–136, 1985.

- 18 Oliver Matz and Andreas Potthoff. Computing small finite nondeterministic automata. In U. H. Engberg, K. G. Larsen, and A. Skou, editors, *Proceedings of the Workshop on Tools and Algorithms for Construction and Analysis of Systems*, BRICS Note Series, pages 74–88. BRICS, 1995.
- 19 Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959.
- 20 Caleb Stanford, Margus Veanes, and Nikolaj Bjørner. Symbolic boolean derivatives for efficiently solving extended regular expression constraints. In Stephen N. Freund and Eran Yahav, editors, *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20–25, 2021*, pages 620–635. ACM, 2021.
- 21 Hellis Tamm. New interpretation and generalization of the Kameda-Weiner method. In *43rd Int. Colloq. on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz Int. Proc. in Informatics (LIPIcs)*, pages 116:1–116:12, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- 22 Hellis Tamm and Brink van der Merwe. Lower bound methods for the size of nondeterministic finite automata revisited. In *Language and Automata Theory and Applications – 11th International Conference, LATA 2017, Umeå, Sweden, March 6-9, 2017, Proceedings*, volume 10168 of *Lecture Notes in Computer Science*, pages 261–272, 2017.
- 23 Brink van der Merwe, Hellis Tamm, and Lynette van Zijl. Minimal DFA for symmetric difference NFA. In *Descriptive Complexity of Formal Systems – 14th International Workshop, DCFS 2012, Braga, Portugal, July 23-25, 2012. Proceedings*, volume 7386 of *Lecture Notes in Computer Science*, pages 307–318. Springer, 2012.
- 24 Lynette van Zijl. On binary \oplus -nfas and succinct descriptions of regular languages. *Theor. Comput. Sci.*, 328(1-2):161–170, 2004.

The Gödel Fibration

Davide Trotta ✉ 

Department of Computer Science, University of Pisa, IT

Matteo Spadetto ✉

School of Mathematics, University of Leeds, UK

Valeria de Paiva ✉

Topos Institute, Berkeley, CA, USA

Abstract

We introduce the notion of a Gödel fibration, which is a fibration categorically embodying both the logical principles of traditional Skolemization (we can exchange the order of quantifiers paying the price of a functional) and the existence of a prenex normal form presentation for every logical formula. Building up from Hofstra’s earlier fibrational characterization of de Paiva’s categorical Dialectica construction, we show that a fibration is an instance of the Dialectica construction if and only if it is a Gödel fibration. This result establishes an intrinsic presentation of the Dialectica fibration, contributing to the understanding of the Dialectica construction itself and of its properties from a logical perspective.

2012 ACM Subject Classification Theory of computation → Higher order logic; Theory of computation → Proof theory; Theory of computation → Constructive mathematics

Keywords and phrases Dialectica category, Gödel fibration, Pseudo-monad

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.87

Related Version The Gödel fibration

Extended Version: <https://arxiv.org/abs/2104.14021>

Funding *Davide Trotta:* the research is supported by the MIUR PRIN 2017FTXR “IT-MaTTerS”.

Matteo Spadetto: the research is supported by a School of Mathematics Full-Time EPSRC Doctoral Training Partnership Studentship.

Valeria de Paiva: the author acknowledges support from AFOSR grant FA9550-20-10348.

1 Introduction

Historically, the Dialectica interpretation was devised by Gödel [10] to prove the (relative) consistency of arithmetic. The interpretation allowed him to reduce the problem of proving the consistency of first-order arithmetic to the problem of proving the consistency of a simply-typed system of computable functionals, the well-known *System T*. The key feature of the translation is that it (mostly) constructively turns formulae of arbitrary quantifier complexity into formulae of the form $\exists x \forall y \alpha(x, y)$.

Over the years, several authors have explained the Dialectica interpretation in categorical terms. In particular, de Paiva [7] introduced the notion of *Dialectica categories* as an internal version of Gödel’s Dialectica Interpretation. The idea is to construct a category $\mathfrak{Dial}(\mathbb{C})$ from a category \mathbb{C} with finite limits. The main focus in de Paiva’s original work is on the categorical structure of the category $\mathfrak{Dial}(\mathbb{C})$ obtained, as this notion of a Dialectica category turns out to be also a model of Girard’s Linear Logic [9].

This construction was first generalized by Hyland, who investigated the Dialectica construction associated to a fibred preorder [12]. Later Biering in her PhD work [3] studied the Dialectica construction for an arbitrary cloven fibration.



© Davide Trotta, Matteo Spadetto, and Valeria de Paiva;
licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 87; pp. 87:1–87:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Meanwhile Hofstra [11] wrote an exposition and interpretation of the Dialectica construction from a modern categorical perspective, emphasizing its universal properties. His work gives centre stage to the well-known concepts of pseudo-monads, simple products and co-products. We take Hofstra’s work as the basis for our work here.

Hofstra shows that the original Dialectica construction \mathfrak{Dial} can be seen as the composition of two free constructions \mathfrak{Sum} and \mathfrak{Prod} , which are the simple sum (or co-product) and product completions, respectively. These completions are fully dual, so we only need to study one and can then deduce results for the other construction. However the whole Dialectica construction is not fully dual, as indicated by the order of the composition of the completions. Our work explains when the Dialectica construction can be performed, which hypotheses are necessary for the categorical construction, which properties of the construction are preserved and why. Most importantly we are able to connect these preservation properties to the logic of the original interpretation, leading up to the definition of what we call a Gödel fibration.

Our contributions

The main contributions of this paper are the following.

1. *We formalize the notion of fibrational quantifier-free formula.* Given Hofstra’s characterization it is clear that instances of the Dialectica construction should have simple products and co-products, as the construction introduces completions under these. What else is necessary to get a Dialectica construction? The first novelty of this work is the characterization of “covering quantifier-free objects” of a fibration. These objects correspond to formulas in the logic system that are quantifier-free. As usually happens in a categorical framework, a syntactical notion of “being quantifier-free” needs to be formalized in terms of a universal property. The logical intuition behind our definition, is that an element α of a fibration \mathfrak{p} is called *quantifier-free* if it satisfies the following universal property, expressed in the internal language of \mathfrak{p} : if there is a proof π of a statement $\exists i\beta(i)$ assuming α , then there exists a *witness* t , which depends on the proof π , together with a proof of $\beta(t)$. Moreover, this must hold for every re-indexing $\alpha(f)$, because in logic if $\alpha(x)$ is quantifier-free then $\alpha(x)[f/x] = \alpha(f)$ is quantifier-free too. The covering requirement, as usual, means that, for every formula of the form $i : I \mid \alpha(i)$, there exists a formula $\beta(i, a, b)$ quantifier-free that is provably equivalent to it $\alpha(i) \dashv\vdash \exists a\forall b\beta(i, a, b)$. Notice that these requirements reflect Gödel’s original translation and, at the same time, they recall standard conditions used in category theory to say that a category is *free* for a given structure. One could think for example about the condition of having enough projectives in the exact completion of Carboni [5].
2. *We introduce the notion of a Gödel fibration.* A Gödel fibration is a fibration with simple products and simple co-products, which, most importantly, admits a class of formal sub-objects which are *free from products and co-products* and *cover* all the elements of the fibre. Then we prove that a Gödel fibration is a fibration categorically embodying both the logical principle of traditional Skolemization and the existence of a prenex normal presentation for every logical formula.
3. *We provide an intrinsic presentation of the Dialectica fibration.* We prove that a given fibration is an instance of the Dialectica construction if and only if it is a Gödel fibration. This result helps understanding the existing notion of Dialectica fibration from a logical perspective because it shows which properties an arbitrary fibration should satisfy to be an instance of the Dialectica construction. In other words, given a fibration \mathfrak{p} there exists a fibration $\bar{\mathfrak{p}}$ such that $\mathfrak{p} \cong \mathfrak{Dial}(\bar{\mathfrak{p}})$ if and only if \mathfrak{p} is a Gödel fibration. From a categorical perspective, we have classified the free-algebras for the Dialectica pseudo-monad.

4. We prove that fibrations associated to the Dialectica construction satisfy a strong constructive feature in terms of witnesses. We have shown that in the internal language of, say Hofstra’s Dialectica fibration $\mathfrak{Dial}(\mathfrak{p})$, i.e. in the logic theory that canonically corresponds to this categorical notion, if there is a proof π of a statement $\exists i \alpha(i)$, then there exists a *witness* t , which depends on the proof π , together with a proof of $\alpha(t)$. This principle is sometimes called the Rule of Choice. For example, Regular Logic (<https://ncatlab.org/nlab/show/regular+logic>) satisfies this principle, see [24].

Related work

In the present paper we provide an intrinsic characterization of the free algebras of the pseudo-monad $\mathfrak{Dial}(-)$ introduced by Hofstra in [11], i.e. we provide necessary and sufficient conditions for a (cloven and split) fibration to be of the form $\mathfrak{Dial}(\mathfrak{p})$ for some fibration \mathfrak{p} . Hofstra’s categorical presentation of the Dialectica construction generalizes to the fibrational setting the original construction introduced by de Paiva [7]. In particular, we recall the structural analysis due to Hyland [12] and Biering [2], where the first fibrational presentations of the Dialectica construction were introduced. For a complete presentation of the theory of fibrations and its connection to type theory, we refer the reader to Jacobs [13], and to [4] for an introduction to pseudomonads.

More recently, modern reformulations of the Dialectica interpretation based on the linearized version of de Paiva have been introduced, aiming to provide categorical models for type theory. A relevant example of this line of work is Moss and von Glehn [17], where the authors are interested not in the original construction, but in a modified version of Gödel’s Dialectica interpretation for models of intensional Martin-Löf type-theory, using the notion of fibred display map category. Their work focus on the preservation of the type constructors, while they drop the layer of predicates from their Dialectica propositions, considering only those Dialectica propositions of the form $\exists x \forall y \top$. In fact, they call their construction the *polynomial model*, explaining that this name fits better, because they are considering the *predicate-free* Dialectica construction. On a similar line, we mention the work of Pédrot [18], investigating the validity of a Dialectica-like construction in a dependent setting. Different variations of the Dialectica interpretation have been devised for automata, e.g. the work of Pradic and Riba [19].

Finally, Topos- and tripos-theoretic versions of the Dialectica construction have been studied by Biering in [3], while the recent work of Shulman [20], describes a “polycategorical” version of a generalization of the Dialectica construction. Other applications of completions involving universal and existential quantifiers can be found in [24, 23, 8], where similar constructions are presented in the language of doctrines.

2 Revisiting categorical quantifiers

One of the pillars of categorical logic is Lawvere’s crucial intuition which considers logical languages and theories as indexed categories and studies their 2-categorical properties. In this setting connectives and quantifiers are characterized in terms of adjointness relations [14, 15, 16].

In this fibrational setting, the intuition is that the base category \mathbf{B} of a fibration $\mathfrak{p}: \mathbf{E} \rightarrow \mathbf{B}$ represents the category of (type-theoretical) contexts, a fibre \mathbf{E}_I represents the propositions $\alpha(i)$ in the context I , and the morphisms are proofs. Cartesian morphisms of \mathfrak{p} induce a *re-indexing* or substitution operation. From this perspective, the simple form of quantification

is described in terms of adjoints to weakening functors π^* along projections π . For example, existential quantification is given by an operation $\coprod_{\pi^*} : \mathbf{E}_{A \times B} \rightarrow \mathbf{E}_A$, which sends a proposition $\alpha(a, b)$ to $\exists b \alpha(a, b)$.

Now we briefly recall the formal definition of a fibration with simple products (or simple universal quantification) and coproducts (or simple existential quantification). For a complete presentation of the theory of fibrations and its connection to type theory, we refer the reader to Jacobs [13]. In this work, we will assume that a fibration \mathbf{p} is always *cloven and split*, i.e. that the re-indexing operation is functorial (these definitions can be found in pages 47 and 49 of [13]).

► **Definition 2.1.** *We say a fibration $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$ over a category \mathbf{B} with finite products has **simple coproducts** when the weakening functors π^* have left adjoints \coprod_{π} satisfying the Beck-Chevalley Condition (abbreviated as *BCC*), i.e. for every pullback square of the form*

$$\begin{array}{ccc} I \times X & \xrightarrow{\pi_I} & I \\ f \times \text{id} \downarrow & & \downarrow f \\ J \times X & \xrightarrow{\pi_J} & J \end{array}$$

the canonical natural transformation $f^* \coprod_{\pi_J} \Rightarrow \coprod_{\pi_I} (f \times \text{id})^*$ is an isomorphism.

Dually, we say that a fibration $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$ has **simple products** when the weakening functors π^* have right adjoints \prod_{π} satisfying *BCC*.

For more details about the notion of fibration having simple coproducts (or simple products) we refer to [13, Def. 1.9.1].

When one deals with quantification, for example in first-order logic, it is very common to assert something like *a formula α is quantifier-free*. This assertion has a natural meaning from a syntactic point of view, but it is not clear how it should be presented from a categorical perspective. The aim of the following definitions, which are generalizations of definitions in [24] to the fibrational setting, is to capture the common property of those elements of a given fibration $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$ which will appear as quantifier-free propositions in the internal language of the fibration \mathbf{p} . We start by defining when an element of a fibre of \mathbf{p} is free from the existential quantifier, and then we dualize the definition for the universal quantifier. (Recall that the symbols \coprod and \prod represent the logical quantifiers \exists and \forall .)

The logical intuition behind the next definition is that an element α is *existential-free* if it satisfies the following universal property: if there is a proof π of a statement $\exists i \beta(i)$ assuming α , then there exists a *witness* t , which depends on the proof π , together with a proof of $\beta(t)$. Moreover, we require that this holds for every re-indexing $\alpha(f)$ because in logic quantifier-free propositions are stable under substitution, i.e. if $\alpha(x)$ is quantifier-free then $\alpha(f)$ is quantifier-free.

► **Definition 2.2.** *Let $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$ be a fibration with simple coproducts. An object α of the fibre \mathbf{E}_I is said to be **\coprod -quantifier-free** if it enjoys the following universal property. For every arrow f and every projection π_A in \mathbf{B} as follows:*

$$A \times B \xrightarrow{\pi_A} A \xrightarrow{f} I$$

and every vertical arrow:

$$f^* \alpha \xrightarrow{h} \prod_{\pi_A} \beta$$

of \mathbf{E}_A , where β is an object of the fibre $\mathbf{E}_{A \times B}$, there exist a unique arrow $A \xrightarrow{g} B$ of \mathbf{B} and a unique vertical arrow $f^* \alpha \xrightarrow{\bar{h}} \langle 1_A, g \rangle^* \beta$ of \mathbf{E}_A such that:

$$h = \left(f^* \alpha \xrightarrow{\bar{h}} \langle 1_A, g \rangle^* \beta \xrightarrow{\langle 1_A, g \rangle^* \eta_\beta} \langle 1_A, g \rangle^* \left(\prod_{\pi_A}^* \beta \right) = \prod_{\pi_A} \beta \right)$$

where $\beta \xrightarrow{\eta_\beta} \prod_{\pi_A}^* \beta$ is the unit at β of the adjunction $\prod_{\pi_A} \dashv \prod_{\pi_A}^*$.

Clarifying the concrete meaning of Definition 2.2, the given object α of \mathbf{E}_I represents a formula $\alpha(i)$. Given an arrow $A \xrightarrow{f} I$ a term $f(a) : I$ is in the context $a : A$, and it is the case that $f^* \alpha$ represents the corresponding formula $\alpha(f(a))$. The object β of $\mathbf{E}_{A \times B}$ corresponds to a formula $\beta(a, b)$, the object $\prod_{\pi_A} \beta$ represents the formula $(\exists b)\beta(a, b)$, which is in the same context $a : A$ of $\alpha(f(a))$. Meanwhile, the object $\langle 1_A, g \rangle^* \beta$ is again the re-indexing of $\beta(a, b)$ through an arrow $A \xrightarrow{\langle 1_A, g \rangle} A \times B$, hence it represents the formula $\beta(a, g(a))$, which is in the same context $a : A$ of $\alpha(f(a))$ and $(\exists b)\beta(a, b)$.

Thus the property we require of the formula $\alpha(i)$ is the following: whenever there is a proof (an arrow h of the fibre) of $(\exists b)\beta(a, b)$ from $\alpha(f(a))$ (for some term $f(a) : I$ in the context $a : A$), then there is a unique term $g(a) : B$ in the context $a : A$ together with a unique proof \bar{h} of $\beta(a, g(a))$ from $\alpha(f(a))$, in such a way that, adding at the end of the proof \bar{h} the canonical proof of $(\exists b)\beta(a, b)$ from $\beta(a, g(a))$ (which is represented by the re-indexing of the unit at β of the adjunction $\prod_{\pi_A} \dashv \prod_{\pi_A}^*$), we get back to the proof h itself of $(\exists b)\beta(a, b)$ from $\alpha(f(a))$. The uniqueness requirement of the term and the proof is due to the proof-relevant nature of fibrations.

Observe that this is precisely the universal property, that we presented before Definition 2.2, enjoyed by a formula which is free from existential quantification.

► **Remark 2.3.** Notice that if we consider a fibration \mathbf{p} with simple coproducts, then one can define a sub-fibration $\mathbf{p}' \rightarrow \mathbf{p}$ such that the fibres of \mathbf{p}' are given by \prod -quantifier-free objects, and the base category of \mathbf{p}' is the same of \mathbf{p} . This follows since \prod -quantifier-free objects are stable under re-indexing by definition.

The next concept we are going to need in the categorical setting reminds us of the *existence of a prenex normal form* in logic. Recall, for example from [6], that in (classical) first-order logic (FOL) every formula is equivalent to some formula in prenex normal form.

► **Definition 2.4.** We say that a fibration with simple coproducts $\mathbf{p} : \mathbf{E} \rightarrow \mathbf{B}$ has **enough \prod -quantifier-free objects** if, for every object I of \mathbf{B} and for every element $\alpha \in \mathbf{E}_I$, there exist an object A and a \prod -quantifier-free object β in $\mathbf{E}_{I \times A}$ such that $\alpha \cong \prod_{\pi_I} \beta$.

By duality we can define the same concept with respect to the universal quantifier.

► **Definition 2.5.** Let $\mathbf{p} : \mathbf{E} \rightarrow \mathbf{B}$ be a fibration with simple products. An object α of the fibre \mathbf{E}_I is said to be **\prod -quantifier-free** if it enjoys the following universal property: for every arrow f and every projection π_A in \mathbf{B} as follows:

$$A \times B \xrightarrow{\pi_A} A \xrightarrow{f} I$$

and every vertical arrow:

$$\prod_{\pi_A} \beta \xrightarrow{h} f^* \alpha$$

87:6 The Gödel Fibration

of \mathbf{E}_A , where β is an object of the fibre $\mathbf{E}_{A \times B}$, there exist a unique arrow $A \xrightarrow{g} B$ of \mathbf{B} and a unique vertical arrow $\langle 1_A, g \rangle^* \beta \xrightarrow{\bar{h}} f^* \alpha$ of \mathbf{E}_A such that:

$$h = \left(\prod_{\pi_A} \beta = \langle 1_A, g \rangle^* (\pi_A^* \prod_{\pi_A} \beta) \xrightarrow{\langle 1_A, g \rangle^* \varepsilon_\beta} \langle 1_A, g \rangle^* \beta \xrightarrow{\bar{h}} f^* \alpha \right)$$

where $\pi_A^* \prod_{\pi_A} \beta \xrightarrow{\varepsilon_\beta} \beta$ is the counit at β of the adjunction $\pi_A^* \dashv \prod_{\pi_A}$.

► **Definition 2.6.** We say that a fibration with simple products $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$ has **enough- \prod -quantifier-free objects** if, for every object I of \mathbf{B} and for every element $\alpha \in \mathbf{E}_I$, there exist an object A and a \prod -quantifier-free object β in $\mathbf{E}_{I \times A}$ such that $\alpha \cong \prod_{\pi_I}(\beta)$.

Now we can introduce a particular kind of fibration called a *Skolem fibration*. The name is chosen because these fibrations satisfy a version of the traditional principle of *Skolemization*, as presented in [10] and [11].

► **Definition 2.7.** A fibration $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$ is called a **Skolem fibration** if:

- its base category \mathbf{B} is cartesian closed;
- the fibration \mathbf{p} has simple products and simple coproducts;
- the fibration \mathbf{p} has enough \prod -quantifier-free objects.
- \prod -quantifier-free objects are stable under simple products, i.e. if $\alpha \in \mathbf{E}_I$ is a \prod -quantifier-free object, then $\prod_{\pi}(\alpha)$ is a \prod -quantifier-free object for every projection π from I .

Notice that the last point of Definition 2.7 implies that, given a Skolem fibration $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$, the sub-fibration $\mathbf{p}': \mathbf{E}' \rightarrow \mathbf{B}$ of \prod -quantifier-free objects of \mathbf{p} defined in Remark 2.3 has simple products.

► **Proposition 2.8 (Skolemization).** Every Skolem fibration \mathbf{p} validates the principle:

$$\forall x \exists y \alpha(i, x, y) \cong \exists f \forall x \alpha(i, x, fx).$$

Proof. Let us consider an element $\alpha \in E_{A_1 \times A_2 \times B}$ and a \prod -quantifier-free object $\gamma \in E_{A_1}$. Hence, for every arrow $\pi_1^*(\gamma) \xrightarrow{h} \prod_{\langle \pi_1, \pi_2 \rangle}(\alpha)$, there is a unique pair (g, \bar{h}) where $A_1 \times A_2 \xrightarrow{g} B$ and $\pi_1^*(\gamma) \xrightarrow{\bar{h}} \langle \pi_1, \pi_2, g \rangle^*(\alpha)$. Since \mathbf{B} has exponents, then we have that g induces a unique arrow $A_1 \xrightarrow{m} B^{A_2}$ such $\langle \pi_1, \pi_2, g \rangle = \langle \pi_1, \pi_2, \text{ev} \langle \pi_1, \pi_2 \rangle \rangle \langle \pi_1, \pi_2, m \pi_1 \rangle$. Therefore we have an arrow

$$\pi_1^*(\gamma) \xrightarrow{\bar{h}} \langle \pi_1, \pi_2, m \pi_1 \rangle^*(\langle \pi_1, \pi_2, \text{ev} \langle \pi_1, \pi_2 \rangle \rangle(\alpha)).$$

Since \mathbf{p} has simple products, \bar{h} induces a unique arrow

$$\gamma \xrightarrow{\bar{\bar{h}}} \prod_{\pi_1} \langle \pi_1, \pi_2, m \pi_1 \rangle^*(\langle \pi_1, \pi_2, \text{ev} \langle \pi_1, \pi_2 \rangle \rangle(\alpha)).$$

Notice that the following square

$$\begin{array}{ccc} A_1 \times A_2 & \xrightarrow{\pi_1} & A_1 \\ \langle \pi_1, \pi_2, m \pi_1 \rangle \downarrow & & \downarrow \langle id_{A_1}, m \rangle \\ A_1 \times A_2 \times B^{A_2} & \xrightarrow{\langle \pi_1, \pi_3 \rangle} & A_1 \times B^{A_2} \end{array}$$

is a pullback, hence by the BCC we have that $\prod_{\pi_1} \langle \pi_1, \pi_2, m\pi_1 \rangle^* \cong \langle \text{id}_{A_1}, m \rangle^* \prod_{\langle \pi_1, \pi_3 \rangle}$. Thus, we get that an arrow f induces a unique pair of arrows (m, \bar{h}) , but again (since \mathfrak{p} has enough \prod -quantifier-free objects) this pair represents a unique arrow of the fibre $E_{A_1}(\gamma, \prod_{\pi_3} \prod_{\langle \pi_1, \pi_1 \rangle} (\langle \pi_1, \pi_2, \text{ev}\langle \pi_1, \pi_2 \rangle \rangle(\alpha)))$, i.e. the fibre

$$E_{A_1 \times A_2}(\pi_1^*(\gamma), \prod_{\langle \pi_1, \pi_2 \rangle} (\alpha))$$

is isomorphic to

$$E_{A_1}(\gamma, \prod_{\pi_1} \prod_{\langle \pi_1, \pi_3 \rangle} (\langle \pi_1, \pi_2, \text{ev}\langle \pi_1, \pi_2 \rangle \rangle(\alpha)))$$

and this means exactly that

$$\prod_{\pi_1} \prod_{\langle \pi_1, \pi_2 \rangle} (\alpha) \cong \prod_{\pi_1} \prod_{\langle \pi_1, \pi_3 \rangle} (\langle \pi_1, \pi_2, \text{ev}\langle \pi_1, \pi_2 \rangle \rangle(\alpha)).$$

The proof for the general case where γ is a generic element of the fibre and not a \prod -quantifier-free object, follows by the observation that the arrows $\pi_1^*(\gamma) \rightarrow \beta$ are in bijection with those of the form $\pi_1^*(\gamma') \rightarrow \prod_{\pi_2} \beta$ for appropriate projections, and where γ' is the \prod -quantifier-free element which covers γ . ◀

Combining Definitions 2.4, 2.6 and 2.7, we introduce the notion of a *Gödel fibration*. The idea is that a Gödel fibration is a Skolem fibration, such that every formula $\alpha(i)$ is equivalent to a formula in *prenex normal form* with respect to \mathfrak{p} , i.e. there exists a formula $\beta(x, y, i)$ free from quantifiers, such that $\alpha(i) \cong \exists x \forall y \beta(x, y, i)$.

► **Definition 2.9.** A Skolem fibration $\mathfrak{p}: E \rightarrow B$ is called a **Gödel fibration** if the sub-fibration $\mathfrak{p}': E' \rightarrow B$, whose elements are \prod -quantifier-free objects, has enough \prod -quantifier-free objects.

► **Remark 2.10.** Observe that if we consider a Gödel fibration $\mathfrak{p}: E \rightarrow B$, an element which is a \prod -quantifier-free object in the sub-fibration \mathfrak{p}' could not be \prod -quantifier-free object of the Gödel fibration. This because in Definition 2.9 of Gödel fibration, the universal property of being a \prod -quantifier-free object is required to hold only with respect to the \prod -quantifier-free objects of \mathfrak{p} .

The following proposition is an immediate consequence of Definition 2.9.

► **Proposition 2.11** (Prenex normal form). In a Gödel fibration $\mathfrak{p}: E \rightarrow B$, for every element α of a fibre E_I there exists an element β such that

$$\alpha(i) \cong \exists x \forall y \beta(x, y, i)$$

and β is \prod -quantifier-free in the sub-fibration \mathfrak{p}' of \prod -quantifier-free objects of \mathfrak{p} .

Proof. Let us consider an element α of the fibre E_I . Since \mathfrak{p} is a Gödel fibration, hence in particular a Skolem fibration, the fibration \mathfrak{p} has enough \prod -quantifier-free objects, and hence there exists an element γ in the fibre $E_{I \times X}$ such that $\alpha \cong \prod_{\pi_I}(\gamma)$. Therefore, since the sub-fibration \mathfrak{p}' has enough \prod -quantifier-free objects, there exists a \prod -quantifier-free object β of \mathfrak{p}' in the fibre $E_{I \times X \times Y}$ such that $\gamma \cong \prod_{\pi_X}(\beta)$, and hence $\alpha \cong \prod_{\pi_I} \prod_{\pi_X}(\beta)$. ◀

3 The Dialectica monad

In this section we assume that $p: E \rightarrow B$ is a cloven and split fibration whose base category B has finite products. First we recall from Hofstra's [11] the free construction $\mathfrak{S}um(-)$ which adds simple sums (or coproducts) to a fibration, and then the dual construction $\mathfrak{P}rod(-)$ which freely adds simple products. Then, we present the Dialectica construction $\mathfrak{D}ial(-)$ and its decomposition in terms of simple coproducts and products completions.

Simple coproducts completion. The category $\mathfrak{S}um(p)$ has:

- as **objects** triples (I, X, α) , where I and X are objects of the base category B and α is an object of the fibre $E_{I \times X}$;
- as **morphisms** triples $(I, X, \alpha) \xrightarrow{(f_0, f_1, \phi)} (J, Y, \beta)$, where $I \xrightarrow{f_0} J$ and $I \times X \xrightarrow{f_1} Y$ are arrows of B and $\alpha(i, x) \xrightarrow{\phi} \beta(f_0(i), f_1(i, x))$ is a morphism of the fibre category $E_{I \times X}$.

The category $\mathfrak{S}um(p)$ is fibred over B via the first component projection and this fibration is denoted by $\mathfrak{S}um(p): \mathfrak{S}um(p) \rightarrow B$. This fibration is called the *simple coproduct (or sum) completion* of p . The intuition behind this definition is that an object (I, X, α) of the fibre category $\mathfrak{S}um(p)_I$ represents a formula $(\exists x : X)\alpha(i, x)$. The assignment $p \mapsto \mathfrak{S}um(p)$ extends to a KZ pseudo-monad on the 2-category of cloven split fibrations, see [11, Theorem 3.9].

► **Remark 3.1 (A presentation of $\mathfrak{S}um(p)$ reindexing functors).** Let $p: E \rightarrow B$ be a cloven and split fibration. Let $I \xrightarrow{f} J$ be an arrow of B and let (J, Y, β) be an object of $\mathfrak{S}um(p)_J$. Then the triple:

$$(I \xrightarrow{f} J, I \times Y \xrightarrow{\pi_Y} Y, \langle f\pi_I, \pi_Y \rangle^* \beta \xrightarrow{1_{\langle f\pi_I, \pi_Y \rangle^* \beta}} \langle f\pi_I, \pi_Y \rangle^* \beta)$$

is $\mathfrak{S}um(p)$ -cartesian $(I, Y, \langle f\pi_I, \pi_Y \rangle^* \beta) \rightarrow (J, Y, \beta)$ over $I \xrightarrow{f} J$. In particular $\mathfrak{S}um(p)$ is endowed with a cloven and split structure. If:

$$(J, Y, \beta) \xrightarrow{(J \times Y \xrightarrow{g} Y', \beta \xrightarrow{\gamma} \langle \pi_J, g \rangle^* \beta')} (J, Y', \beta')$$

is an arrow of $\mathfrak{S}um(p)_J$ (observe the omission of the first component, as it is forced to be the identity arrow on J) then its f -reindexing is the pair:

$$(I, Y, \langle f\pi_I, \pi_Y \rangle^* \beta) \xrightarrow{(g\langle f\pi_I, \pi_Y \rangle, \langle f\pi_I, \pi_Y \rangle^* \gamma)} (I, Y', \langle f\pi_I, \pi_{Y'} \rangle^* \beta')$$

of $\mathfrak{S}um(p)_I$, whose first component is the arrow $I \times Y \xrightarrow{g\langle f\pi_I, \pi_Y \rangle} Y'$ of B and whose second one is the arrow:

$$\langle f\pi_I, \pi_Y \rangle^* \beta \xrightarrow{\langle f\pi_I, \pi_Y \rangle^* \gamma} \langle f\pi_I, \pi_Y \rangle^* \langle \pi_J, g \rangle^* \beta' = \langle \pi_I, g\langle f\pi_I, \pi_Y \rangle \rangle^* \langle f\pi_I, \pi_{Y'} \rangle^* \beta'$$

of $E_{I \times Y}$.

Now, let us assume that f is a projection $J \times K \xrightarrow{\pi_J} J$. In this particular case (in which we are mostly interested) such an annoying presentation *collapses* into the following easier one: the π_J -weakening of the arrow (g, γ) of $\mathfrak{S}um(p)_J$ is the pair:

$$(J \times K, Y, \langle \pi_J, \pi_Y \rangle^* \beta) \xrightarrow{(g\langle \pi_J, \pi_Y \rangle, \langle \pi_J, \pi_Y \rangle^* \gamma)} (J \times K, Y', \langle \pi_J, \pi_{Y'} \rangle^* \beta')$$

of $\mathfrak{S}um(p)_I$, whose first component is the arrow $J \times K \times Y \xrightarrow{g\langle \pi_J, \pi_Y \rangle} Y'$ of B and whose second one is the arrow:

$$\langle \pi_J, \pi_Y \rangle^* \beta \xrightarrow{\langle \pi_J, \pi_Y \rangle^* \gamma} \langle \pi_J, \pi_Y \rangle^* \langle \pi_J, g \rangle^* \beta' = \langle \langle \pi_J, \pi_K \rangle, g\langle \pi_J, \pi_Y \rangle \rangle^* \langle \pi_J, \pi_{Y'} \rangle^* \beta'$$

of $E_{J \times K \times Y}$.

► **Remark 3.2** ($\mathfrak{S}\text{um}(\mathfrak{p})$ has simple coproducts). Let \mathfrak{p} be a cloven and split fibration and let us consider a projection $J \times K \xrightarrow{\pi_J} J$ of \mathbf{B} . The left adjoint \coprod_{π_J} of the π_J -weakening π_J^* in $\mathfrak{S}\text{um}(\mathfrak{p})$ exists and sends an arrow:

$$(J \times K, Y, \beta) \xrightarrow{(J \times K \times Y \xrightarrow{g} Y', \beta \xrightarrow{\gamma} \langle (\pi_J, \pi_K), g \rangle^* \beta')} (J \times K, Y', \beta')$$

of $\mathfrak{S}\text{um}(\mathfrak{p})_{J \times K}$ to the arrow:

$$(J, K \times Y, \beta) \xrightarrow{(J \times K \times Y \xrightarrow{\langle \pi_K, g \rangle} K \times Y', \beta \xrightarrow{\gamma} \langle \pi_J, \langle \pi_K, g \rangle \rangle^* \beta')} (J, K \times Y', \beta')$$

of $\mathfrak{S}\text{um}(\mathfrak{p})_J$, which we also denote as:

$$\coprod_{\pi_J} (J \times K, Y, \beta) \xrightarrow{\coprod_{\pi_J} (g, \gamma)} \coprod_{\pi_J} (J \times K, Y', \beta').$$

► **Remark 3.3.** Let $\mathfrak{p}: \mathbf{E} \rightarrow \mathbf{B}$ be a fibration and consider its simple coproduct completion $\mathfrak{S}\text{um}(\mathfrak{p}): \mathfrak{S}\text{um}(\mathfrak{p}) \rightarrow \mathbf{B}$. As a consequence of Remark 3.2, every element (I, A, α) of the fibre $\mathfrak{S}\text{um}(\mathfrak{p})_I$ is isomorphic to an object of the form $\coprod_{\pi_I} (I \times A, 1, \alpha')$.

Notice that, by dualising the previous construction, one gets the notion of simple product completion together with its analogous version of the previous characterization.

Simple products completion. The category $\mathfrak{P}\text{rod}(\mathfrak{p})$ is the one:

- whose **objects** are triples (I, X, α) , where I and X are objects of the base category \mathbf{B} and α is an object of the fibre $\mathbf{E}_{I \times X}$;
 - whose **morphisms** are triples $(I, X, \alpha) \xrightarrow{(f_0, f_1, \phi)} (J, Y, \beta)$, where $I \xrightarrow{f_0} J$ and $I \times Y \xrightarrow{f_1} X$ are arrows of \mathbf{B} and $\alpha(i, f_1(i, y)) \xrightarrow{\phi} \beta(f_0(i), y)$ is a morphism of the fibre category $\mathbf{E}_{I \times X}$.
- Again, the category $\mathfrak{P}\text{rod}(\mathfrak{p})$ is fibred over \mathbf{B} via first component projection and this fibration is denoted by $\mathfrak{P}\text{rod}(\mathfrak{p}): \mathfrak{P}\text{rod}(\mathfrak{p}) \rightarrow \mathbf{B}$ and called **simple product completion** of \mathfrak{p} . The intuition behind this definition is that an object (I, X, α) of the fibre category $\mathfrak{P}\text{rod}(\mathfrak{p})_I$ represents a formula $(\forall x : X)\alpha(i, x)$.

► **Proposition 3.4** (Hofstra [11]). *There is an isomorphism of fibrations:*

$$\mathfrak{P}\text{rod}(\mathfrak{p}) \cong \mathfrak{S}\text{um}(\mathfrak{p}^{\text{op}})^{\text{op}}$$

which is natural in \mathfrak{p} .

Here one has to recall that \mathfrak{p}^{op} stands for the *fibrewise opposite* of \mathfrak{p} , see [13] or [11, Def. 2.8].

Again, the assignment $\mathfrak{p} \mapsto \mathfrak{P}\text{rod}(\mathfrak{p})$ extends to a co-KZ pseudo-monad on the 2-category of cloven split fibrations, and its 2-category of pseudo-algebras is equivalent to the 2-category of fibrations with simple products, see [11, Theorem 3.12].

We conclude this section by recalling the presentation of the Dialectica construction and its presentation via the product-coproduct completions.

Dialectica construction. Let $\mathfrak{p}: \mathbf{E} \rightarrow \mathbf{B}$ be a fibration. Define a category $\mathfrak{D}\text{ial}(\mathfrak{p})$ as follows:

- **objects** are quadruples (I, X, U, α) where I, X and U are objects of the base category \mathbf{B} and $\alpha \in \mathbf{E}_{I \times X \times U}$ is an objects of the fibre of \mathfrak{p} over $I \times X \times U$;
- a **morphism** from (I, X, U, α) to (J, Y, V, β) is a quadruple (f, f_0, f_1, ϕ) where

87:10 The Gödel Fibration

1. $I \xrightarrow{f} J$ is a morphism in \mathbf{B} ;
2. $I \times X \xrightarrow{f_0} Y$ is a morphism in \mathbf{B} ;
3. $I \times X \times V \xrightarrow{f_1} U$ is a morphism in \mathbf{B} ;
4. $\alpha(i, x, f_1(i, x, v)) \xrightarrow{\phi} \beta(f(i), f_0(i, x), v)$ is an arrow in the fibre over $I \times X \times V$.

This is a fibration on \mathbf{B} with the projection on the first component. Hofstra's key observation is that the construction of the fibration $\mathfrak{Dial}(\mathfrak{p})$ can be decomposed in two steps.

► **Lemma 3.5** (Hofstra [11]). *There is an isomorphism of fibrations, natural in \mathfrak{p} :*

$$\mathfrak{Dial}(\mathfrak{p}) \cong \mathfrak{Sum}(\mathfrak{Prod}(\mathfrak{p})).$$

Notice that the pseudo-functor $\mathfrak{Sum}(\mathfrak{Prod}(-))$ is not a pseudo-monad in general, but, in the case the base category \mathbf{B} of a fibration $\mathfrak{p}: \mathbf{E} \rightarrow \mathbf{B}$ is cartesian closed, one can show that there exists a pseudo-distributive law

$$\mathfrak{Prod}\mathfrak{Sum} \xrightarrow{\lambda} \mathfrak{Sum}\mathfrak{Prod}$$

of pseudo-monads, see [11, Theorem 4.4]. Therefore, by the known equivalence between liftings of pseudo-monads and pseudo-distributive laws, see for example [21, 22], in this case we have that $\mathfrak{Sum}(\mathfrak{Prod}(-))$ is a pseudo-monad.

A notably advantage of this algebraic presentation of the dialectica construction, is that the principle of Skolemisation is represented by the pseudo-distributive law λ .

► **Theorem 3.6** (Hofstra [11]). *When the base category \mathbf{B} of a fibration \mathfrak{p} is cartesian closed, the fibration $\mathfrak{Dial}(\mathfrak{p})$ satisfies the principle*

$$\forall x \exists y \alpha(i, x, y) \cong \exists f \forall x \alpha(i, x, fx)$$

for every α .

4 An intrinsic characterization of Dialectica fibrations

The main goal of this section is to connect the notion of Gödel fibration with that of Dialectica construction, proving that a given fibration \mathfrak{p} is an instance of the Dialectica construction, i.e. there exists a fibration \mathfrak{p}' such that $\mathfrak{p} \cong \mathfrak{Dial}(\mathfrak{p}')$, if and only if \mathfrak{p} is a Gödel fibration. This result allows us to give an intrinsic definition of a Dialectica fibration because it shows which *properties* an arbitrary fibration should satisfy to be an instance of the Dialectica construction. Moreover, our proof of this equivalence is constructive, in the sense that when \mathfrak{p} is a Gödel fibration, we are able to explicitly define and construct the fibration \mathfrak{p}' such that $\mathfrak{p} \cong \mathfrak{Dial}(\mathfrak{p}')$.

To show this, we take the advantage of Hofstra's decomposition $\mathfrak{Dial}(-) \cong \mathfrak{Sum}(\mathfrak{Prod}(-))$, and we start by showing how fibrations which are instances of the free construction $\mathfrak{Sum}(-)$ (and $\mathfrak{Prod}(-)$) can be described in terms of fibrations with \coprod -quantifier-free objects (and \prod -quantifier-free objects).

► **Proposition 4.1.** *Let $\mathfrak{p}: \mathbf{E} \rightarrow \mathbf{B}$ be a fibration, and let us consider the simple coproduct completion $\mathfrak{Sum}(\mathfrak{p})$. Let I be an object of \mathbf{B} and let α be an object of its fibre \mathbf{E}_I . Then every object of the form $(I, 1, \alpha)$ in the fibre $\mathfrak{Sum}(\mathfrak{p})_I$ is a \coprod -quantifier-free element of $\mathfrak{Sum}(\mathfrak{p})$. Moreover, the \coprod -quantifier-free objects of $\mathfrak{Sum}(\mathfrak{p})$ are up to isomorphism the elements of the form $(I, 1, \alpha)$.*

Proof. First we prove that every element of the form $(I, 1, \alpha)$ is a \coprod -quantifier-free object. Let us consider an arrow

$$\eta_{\mathfrak{p}}(\alpha) = (I, 1, \alpha) \xrightarrow{(f, \phi)} \coprod_{\pi_I} (I \times A, B, \beta)$$

where $I \xrightarrow{f = \langle g_1, g_2 \rangle} A \times B$. We are going to prove that

$$\eta_{\mathfrak{p}}(\alpha) \xrightarrow{(g_2, \phi)} \langle 1_I, g_1 \rangle^* (I \times A, B, \beta)$$

is an arrow of $\mathfrak{S}\text{um}(\mathfrak{p})_I$ and that $(f, \phi) = (\langle 1_I, g_1 \rangle^* \eta_{\beta})(g_2, \phi)$, where η_{β} is the unit at $(I \times A, B, \beta)$ of the adjunction $\coprod_{\pi_I} \dashv \pi_I^*$.

Moreover, we have to prove that such a choice of arrows $I \xrightarrow{g} A$ of \mathfrak{B} and $\eta_{\mathfrak{p}}(\alpha) \xrightarrow{\bar{h}} \langle 1_I, h \rangle^* (I \times A, B, \beta)$ of $\mathfrak{S}\text{um}(p)_I$ is unique. That is, whenever the equality:

$$(f, \phi) = (\langle 1_I, g \rangle^* \eta_{\beta}) \bar{h}$$

holds, it is the case that $g = g_1$ and $\bar{h} = (g_2, \phi)$.

By Remarks 3.1 and 3.2, it is the case that $\coprod_{\pi_I} (I \times A, B, \beta) = (I, A \times B, \beta)$, and that $\langle 1_I, g_1 \rangle^* (I, A \times B, \beta) = (I, B, \langle \pi_I, g_1 \pi_I, \pi_B \rangle^* \beta)$, where π_I and π_B are the projections from $I \times B$. Then:

$$\eta_{\mathfrak{p}}(\alpha) \xrightarrow{(g_2, \phi)} \langle 1_I, g_1 \rangle^* (I \times A, B, \beta) = (I, B, \langle \pi_I, g_1 \pi_I, \pi_B \rangle^* \beta)$$

is a morphism of $\mathfrak{S}\text{um}(\mathfrak{p})_I$ since $I \xrightarrow{g_2} B$ is an arrow of \mathfrak{B} and:

$$\alpha \xrightarrow{\phi} \langle 1_I, g_2 \rangle^* \langle \pi_I, g_1 \pi_I, \pi_B \rangle^* \beta = \langle 1_I, g_1, g_2 \rangle^* \beta = \langle 1_I, f \rangle^* \beta$$

is a vertical morphism of \mathfrak{E}_I . Observe that η_{β} is the transpose along the adjunction $\coprod_{\pi_I} \dashv \pi_I^*$ of the identity arrow of $(I, A \times B, \beta) = \coprod_{\pi_I} (I \times A, B, \beta)$. Hence η_{β} is the arrow:

$$(I \times A, B, \beta) \xrightarrow{(\pi_{A \times B}, 1_{\beta})} (I \times A, A \times B, \langle \pi_I, \pi_{A \times B} \rangle^* \beta)$$

and its $\langle 1_I, g_1 \rangle$ -reindexing is the arrow:

$$(I, B, \langle \pi_I, g_1 \pi_I, \pi_B \rangle^* \beta) \xrightarrow{(\langle g_1 \pi_I, \pi_B \rangle, 1_{\langle \pi_I, g_1 \pi_I, \pi_B \rangle^* \beta})} (I, A \times B, \beta)$$

whose precomposition by (g_2, ϕ) yields indeed the arrow (f, ϕ) .

Let us assume that the equality:

$$(f, \phi) = (\langle 1_I, g \rangle^* \eta_{\beta}) \bar{h} \tag{1}$$

holds for some arrow $I \xrightarrow{g} A$ of \mathfrak{B} and $\eta_{\mathfrak{p}}(\alpha) \xrightarrow{\bar{h} = (h_2, \psi)} \langle 1_I, h \rangle^* (I \times A, B, \beta)$ of $\mathfrak{S}\text{um}(p)_I$. As it is the case that:

$$\langle 1_I, g \rangle^* \eta_{\beta} = (\langle g \pi_I, \pi_B \rangle, 1_{\langle \pi_I, g \pi_I, \pi_B \rangle^* \beta})$$

one might compute the right-hand side of the equality (1) and infer the equality:

$$(f, \phi) = (\langle g, h_2 \rangle, \psi)$$

which implies that $g = g_1$ and $\bar{h} = (h_2, \psi) = (g_2, \phi)$.

87:12 The Gödel Fibration

Finally, notice that Whenever f is an arrow $A \rightarrow I$ of \mathbf{B} , it is the case that the f -reindexing of $(I, 1, \alpha)$ is the triple $(A, 1, f^*\alpha)$, which is still a *quantifier-free formula*, that is, its second component is terminal in \mathbf{B} .

Conversely, let us assume that the triple (I, A, α) is \coprod -quantifier-free and let us consider its identity arrow $(I, A, \alpha) \rightarrow (I, A, \alpha) = \coprod_{\pi_I} (I \times A, 1, \alpha)$. By \coprod -quantifier-freeness, there are an arrow $I \xrightarrow{g} A$ of \mathbf{B} and an arrow:

$$(I, A, \alpha) \xrightarrow{(I \times A \xrightarrow{1} 1, \alpha \xrightarrow{\phi} \pi_I^* \langle 1_I, g \rangle^* \alpha = \langle \pi_I, g \pi_I \rangle^* \alpha)} \langle 1_I, g \rangle^* (I \times A, 1, \alpha) = (I, 1, \langle 1_I, g \rangle^* \alpha)$$

of $\mathfrak{S}\text{um}(\mathfrak{p})_I$ such that the identity arrow $(\pi_A, 1_\alpha)$ of (I, A, α) equals the composition:

$$\left((I, A, \alpha) \xrightarrow{(I \times A \xrightarrow{1} 1, \phi)} (I, 1, \langle 1_I, g \rangle^* \alpha) \xrightarrow{(g, 1_{\langle 1_I, g \rangle^* \alpha})} (I, A, \alpha) \right)$$

where the pair $(g, 1_{\langle 1_I, g \rangle^* \alpha})$ is nothing but the $\langle 1_I, g \rangle$ -reindexing of the unit at $(I \times A, 1, \alpha)$ of the adjunction $\coprod_{\pi_I} \dashv \pi_I^*$. We infer by this arrow equality that it needs to be the case that $(I \times A \xrightarrow{\pi_A} A) = (I \times A \xrightarrow{\pi_I} I \xrightarrow{g} A)$ and that:

$$(\alpha \xrightarrow{\phi} \langle \pi_I, g \pi_I \rangle^* \alpha = \langle \pi_I, \pi_A \rangle^* \alpha = \alpha \xrightarrow{1_\alpha} \alpha) = 1_\alpha$$

which means that $\phi = 1_\alpha$. Finally we observe that the composition:

$$(I, 1, \langle 1_I, g \rangle^* \alpha) \xrightarrow{(g, 1_{\langle 1_I, g \rangle^* \alpha})} (I, A, \alpha) \xrightarrow{(I \times A \xrightarrow{1} 1, \phi = 1_\alpha)} (I, 1, \langle 1_I, g \rangle^* \alpha)$$

equals the identity arrow $(I \times 1 \xrightarrow{1} 1, 1_{\langle 1_I, g \rangle^* \alpha})$. This concludes that the pair $(I \times A \xrightarrow{1} 1, 1_\alpha)$ is actually an isomorphism $(I, A, \alpha) \cong (I, 1, \langle 1_I, g \rangle^* \alpha)$. \blacktriangleleft

► **Remark 4.2.** Let $\mathfrak{p}: \mathbf{E} \rightarrow \mathbf{B}$ be a fibration and let I be an object of \mathbf{B} . Let us consider an arrow $(I, A, \alpha) \xrightarrow{(f, \phi)} (I, B, \beta)$ of $\mathfrak{S}\text{um}(\mathfrak{p})_I$. W.l.o.f we can assume (I, A, α) to be of the form $\coprod_{\pi_I} (I \times A, 1, \alpha)$, see Remark 3.3, so we might consider its transpose $(I \times A, 1, \alpha) \xrightarrow{(1_{I \times A}, f, \phi)} \pi_I^* (I, B, \beta) = (I \times A, B, \langle \pi_I, \pi_B \rangle^* \beta)$, which is the unique arrow making the following diagram:

$$\begin{array}{ccc} \coprod_{\pi_I} (I \times A, 1, \alpha) & \xrightarrow{(1_{I \times A}, f, \phi)} & (I, B, \beta) \\ \coprod_{\pi_I} (1_{I \times A}, f, \phi) \downarrow & \nearrow \varepsilon_{(I, B, \beta)} & \\ \coprod_{\pi_I} \pi_I^* (I, B, \beta) & & \end{array}$$

commute. Moreover, as $(I \times A, B, \langle \pi_I, \pi_B \rangle^* \beta) = \coprod_{\pi_{I \times A}} (I \times A \times B, 1, \langle \pi_I, \pi_B \rangle^* \beta)$, see Proposition 4.1, the arrow $(1_{I \times A}, f, \phi)$ factors uniquely as the arrow:

$$(I \times A, 1, \alpha) \xrightarrow{(1, \phi)} \langle 1_{I \times A}, f \rangle^* (I \times A \times B, 1, \langle \pi_I, \pi_B \rangle^* \beta) = (I \times A, 1, \langle \pi_I, f \rangle^* \beta)$$

(which can be uniquely expressed as the image $(\bar{\mathfrak{p}} \hookrightarrow \mathfrak{S}\text{um}(\bar{\mathfrak{p}}))\phi$ of the arrow $\alpha \xrightarrow{\phi} \langle \pi_I, f \rangle^* \beta$ of $\bar{E}_{I \times A}$) followed by the arrow:

$$(I \times A, 1, \langle \pi_I, f \rangle^* \beta) \xrightarrow{\langle 1_{I \times A}, f \rangle^* \eta_{(I \times A \times B, 1, \langle \pi_I, \pi_B \rangle^* \beta)}} \coprod_{\pi_{I \times A}} (I \times A \times B, 1, \langle \pi_I, \pi_B \rangle^* \beta)$$

which is the $\langle 1_{I \times A}, f \rangle$ -reindexing of the unit:

$$(I \times A \times B, 1, \langle \pi_I, \pi_B \rangle^* \beta) \xrightarrow{\eta_{(I \times A \times B, 1, \langle \pi_I, \pi_B \rangle^* \beta)}} (I \times A \times B, B, \langle \pi_I, \pi_B \rangle^* \beta)$$

of the adjunction $\coprod_{\pi_{I \times A}} \dashv \pi_{I \times A}^*$.

Notice that in Proposition 4.1 the elements of the form $(I, 1, \alpha)$ represent propositions which are free from the existential quantifier.

► **Remark 4.3.** The analogous of Remark 4.2 can be proved for a fibration having enough \coprod -quantifier-free objects. In other words, in this kind of fibration the arrows of the fibres are completely described by arrows between quantifier-free objects, unit and counit of adjunctions given by coproducts.

Now we have all the tools to give an *intrinsic* description of the free-algebras for the pseudo-monad which adds the simple coproducts to a given fibration.

► **Theorem 4.4.** *A fibration $\mathfrak{p}: \mathbf{E} \rightarrow \mathbf{B}$ with simple coproducts is an instance of simple coproduct completion if and only if it has enough \coprod -quantifier-free objects. Moreover, in this case $\mathfrak{p} \cong \mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p}')$ where \mathfrak{p}' is the subfibration of \coprod -free-quantifiers objects of \mathfrak{p} .*

Proof. We define $\mathfrak{p}': \mathbf{E}' \rightarrow \mathbf{B}$ the full-subfibration of $\mathfrak{p}: \mathbf{E} \rightarrow \mathbf{B}$ such that the objects of \mathbf{E}' are the \coprod -quantifier-free objects. By the universal property of the inclusion morphism $\mathfrak{p}' \hookrightarrow \mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p})$, there is unique a morphism of fibrations with simple coproducts $F: \mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p}') \rightarrow \mathfrak{p}$ commuting with the inclusion morphisms $\mathfrak{p}' \xrightarrow{\eta_{\mathfrak{p}'}} \mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p}')$ and $\mathfrak{p}' \hookrightarrow \mathfrak{p}$. We claim that F is an equivalence of fibrations. Firstly we observe that F is essentially surjective and then we show that it is fully faithful. From now on, whenever π is a projection in \mathbf{B} , we indicate as \coprod_{π} the left adjoint to the π -weakening w.r.t. $\mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p}')$ and as \sum_{π} the one w.r.t. \mathfrak{p} . Observe that:

$$F(I, 1, \gamma) = F(\mathfrak{p}' \xrightarrow{\eta_{\mathfrak{p}'}} \mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p}'))\gamma = (\mathfrak{p}' \hookrightarrow \mathfrak{p})\gamma = \gamma$$

for every I in \mathbf{B} and every γ in \mathbf{E}'_I .

Essential surjectivity. Let α be an object of \mathbf{E} and let I be the object $\mathfrak{p}\alpha$ of \mathbf{B} . Since \mathfrak{p} has enough \coprod -quantifier-free objects, there are J in \mathbf{B} and β in $\mathbf{E}_{I \times J}$ such that $\sum_{\pi_I} \beta \cong \alpha$. Since F preserves simple coproducts, it is the case that:

$$F(I, J, \beta) = F \coprod_{\pi_I} (I \times J, 1, \beta) = \sum_{\pi_I} F(I \times J, 1, \beta) = \sum_{\pi_I} \beta$$

and we are done. Observe that (I, J, β) is an object of \mathbf{E}_I , hence the functor $\mathbf{E}_I \rightarrow \mathbf{E}'_I$ induced by F is essentially surjective as well.

Full faithfulness. It suffices to prove that the morphism F of fibrations over \mathbf{B} gives rise to an equivalence $\mathbf{E}_I \rightarrow \mathbf{E}'_I$, for any given object I of \mathbf{B} (see [13]). As the essential surjectivity of $F \upharpoonright_{\mathbf{E}_I}: \mathbf{E}_I \rightarrow \mathbf{E}'_I$ follows by the previous part, we only need to observe its full faithfulness.

By Remark 4.2 we write a given arrow $(I, A, \alpha) \xrightarrow{(f, \phi)} (I, B, \beta)$ of \mathbf{E}_I as the composition:

$$\varepsilon_{(I, B, \beta)} \left(\coprod_{\pi_I} \langle 1_{I \times A}, f \rangle^* \eta_{(I \times A \times B, 1, \langle \pi_I, \pi_B \rangle^* \beta)} \right) \left(\coprod_{\pi_I} (\mathfrak{p}' \hookrightarrow \mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p}')) \phi \right)$$

and this factorisation is unique, because of the uniqueness of adjoint transposition, because of the uniqueness-part of Proposition 4.1 and because of faithfulness of the functor $\mathfrak{p}' \hookrightarrow \mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p}')$. As F is forced to preserve simple coproducts and commutes with the inclusion morphisms $\mathfrak{p}' \xrightarrow{\eta_{\mathfrak{p}'}} \mathfrak{S}\mathfrak{u}\mathfrak{m}(\mathfrak{p}')$ and $\mathfrak{p}' \hookrightarrow \mathfrak{p}$, the arrow $F(f, \phi)$ equals the composition:

$$\varepsilon_{(\sum_{\pi_I} \beta)} \left(\sum_{\pi_I} \langle 1_{I \times A}, f \rangle^* \eta_{(\pi_I, \pi_B)^* \beta} \right) \left(\sum_{\pi_I} \phi \right)$$

which is indeed an arrow $\sum_{\pi_I} \alpha \rightarrow \sum_{\pi_I} \beta$. Observe that, analogously, every arrow $\sum_{\pi_I} \alpha \rightarrow \sum_{\pi_I} \beta$ of \mathbf{E}'_I can be uniquely factored as such a composition, again by the existence and the uniqueness of the adjoint transposition, by Definition 2.2 (recall that \mathbf{p} is assumed to have \llbracket -quantifier-free objects) and by full faithfulness of $\mathbf{p}' \hookrightarrow \mathbf{p}$. Hence the function:

$$\mathbf{E}_I((I, A, \alpha), (I, B, \beta)) \rightarrow \mathbf{E}'_I\left(\sum_{\pi_I} \alpha, \sum_{\pi_I} \beta\right)$$

induced by $F \upharpoonright_{\mathbf{E}_I}$ is bijective, i.e. $F \upharpoonright_{\mathbf{E}_I}$ is fully faithful. \blacktriangleleft

Notice that the characterization of Theorem 4.4 can be obtained also for the simple product completion because of the equivalence the equivalence $\mathfrak{Prod}(\mathbf{p}) \cong \mathfrak{Sum}(\mathbf{p}^{\text{op}})^{\text{op}}$, see Proposition 3.4.

► **Theorem 4.5.** *A fibration $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$ with simple products is an instance of simple product completion if and only if it has enough \llbracket -quantifier-free objects. Moreover, in this case $\mathbf{p} \cong \mathfrak{Prod}(\mathbf{p}'')$ where \mathbf{p}'' is the subfibration of \llbracket -free-quantifiers objects of \mathbf{p} .*

Proof. It follows by Theorem 4.4 and Proposition 3.4. \blacktriangleleft

Combining Lemma 3.5, Theorem 4.4 and Theorem 4.5 we can prove the main result of our work, which allows us to recognize if an arbitrary fibration \mathbf{p} is an instance of the Dialectica construction or not, and if it is, we can construct the fibration $\bar{\mathbf{p}}$ such that $\mathfrak{Dial}(\bar{\mathbf{p}}) \cong \mathbf{p}$.

► **Theorem 4.6.** *Let $\mathbf{p}: \mathbf{E} \rightarrow \mathbf{B}$ be a fibration with products, coproducts and such that \mathbf{B} is cartesian closed. Then there exists a fibration $\bar{\mathbf{p}}$ such that for $\mathfrak{Dial}(\bar{\mathbf{p}}) \cong \mathbf{p}$ if and only if \mathbf{p} is a Gödel fibration.*

Proof. By Lemma 3.5 we have that $\mathfrak{Dial}(-) \cong \mathfrak{Sum}(\mathfrak{Prod}(-))$. Therefore, the result follows from Theorem 4.5 and Theorem 4.4 by directly rephrasing the sequential application of these results. \blacktriangleleft

► **Remark 4.7.** Notice that from a pure categorical perspective Theorem 4.6 provides a characterization of the free-algebras of the pseudo-monad $\mathfrak{Dial}(-)$.

5 Conclusion

Our results develop the original Dialectica construction from both a categorical and logical perspectives, which contributes to a deeper understanding of the construction.

Our main result Theorem 4.6, provides an internal characterization of fibrations which are instances of the Dialectica construction, highlighting the key features a fibration should satisfy, namely it must be a Gödel fibration, to be an instance of the Dialectica construction.

Our presentation in terms of Gödel fibrations underlines a double nature of Dialectica fibrations: they satisfy principles which are typical of classical logic, such as the existence of a prenex normal form presentation for formulae, but they also satisfy principles normally associated to intuitionistic logic. For example, they satisfy the existence of terms witnessing a proof: for every proof of $\alpha \vdash \exists x \beta(x)$ where α is quantifier-free, we have a proof of $\alpha \vdash \beta(t)$ for some term t .

Dialectica-like constructions are pervasive in several areas of mathematics and computer science, and we briefly describe some future work, based on our previous analysis. We wonder if the decomposition introduced by Hofstra can be extended or modified to provide similar results for *cousins* of the Dialectica construction. In particular, we believe that this decomposition, combined with the results presented in [23], could be generalized to the context of dependent type theory.

There are two fibrations which seem to share common features with the Dialectica construction. In particular, we would like to investigate and compare the fibrations arising from work by Abramsky and Väänänen [1] on the Hodges semantics for independence-friendly logic and the Dialectica tripos, which is a model of separation logic [3].

Finally, the strong constructive features of Dialectica fibrations we have shown suggest that these kinds of fibrations could lead to interesting applications in constructive foundations for mathematics and proof theory.

References

- 1 S. Abramsky and J. Väänänen. From if to bi : A tale of dependence and separation. *Synthese*, 167, February 2011.
- 2 B. Biering. Cartesian closed Dialectica categories. *Annals of Pure and Applied Logic*, 156(2):290–307, 2008.
- 3 B. Biering. Dialectica interpretations – a categorical analysis (PhD Thesis), 2008.
- 4 R. Blackwell, G.M. Kelly, and J. Power. Two-dimensional monad theory. *J. Pure Appl. Algebra*, 59:1–41, 1989.
- 5 A. Carboni and E.V. Vitale. Regular and exact completions. *Journal of Pure and Applied Algebra*, 125(1):79–116, 1998.
- 6 D. Dalen. *Logic and Structure*. Universitext (1979). Springer, 2004.
- 7 V. de Paiva. The dialectica categories. *Categories in Computer Science and Logic*, 92:47–62, 1989.
- 8 J. Frey. *A fibrational study of realizability toposes (PhD Thesis)*. PhD thesis, Universite Paris Diderot – Paris 7, 2014.
- 9 J.Y. Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.
- 10 K. Gödel, S. Feferman, et al. *Kurt Gödel: Collected Works: Volume II: Publications 1938-1974*, volume 2. Oxford University Press, 1986.
- 11 P. Hofstra. The dialectica monad and its cousins. *Models, logics, and higherdimensional categories: A tribute to the work of Mihály Makkai*, 53:107–139, 2011.
- 12 J.M.E. Hyland. Proof theory in the abstract. *Annals of Pure and Applied Logic*, 114(1):43–78, 2002. Troelstra Festschrift.
- 13 B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the foundations of mathematics*. North Holland Publishing Company, 1999.
- 14 F.W. Lawvere. Adjointness in foundations. *Dialectica*, 23:281–296, 1969.
- 15 F.W. Lawvere. Diagonal arguments and cartesian closed categories. In *Category Theory, Homology Theory and their Applications*, volume 2, page 134–145. Springer, 1969.
- 16 F.W. Lawvere. Equality in hyperdoctrines and comprehension schema as an adjoint functor. In A. Heller, editor, *New York Symposium on Application of Categorical Algebra*, volume 2, page 1–14. American Mathematical Society, 1970.
- 17 S.K. Moss and T. von Glehn. Dialectica models of type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 739–748, 2018.
- 18 P.M. Pédrot. A functional functional interpretation. In *CSL-LICS 2014 – Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science*, CSL-LICS '14, New York, NY, USA, 2014. Association for Computing Machinery.

- 19 P. Pradic and C. Riba. A dialectica-like interpretation of a linear mso on infinite words. In Mikołaj Bojańczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures*, pages 470–487, Cham, 2019. Springer International Publishing.
- 20 M. Shulman. The 2-chu-dialectica construction and the polycategory of multivariable adjunctions. *Theory and Applications of Categories*, 35(4):89–136, 2020.
- 21 M. Tanaka. *Pseudo-distributive laws and a unified framework for variable binding*. PhD thesis, The University of Edinburgh, 2004.
- 22 M. Tanaka and J. Power. A unified category-theoretic semantics for binding signatures in substructural logics. *Journal of Logic and Computation*, 16(1), 2006.
- 23 D. Trotta. The existential completion. *Theory and Applications of Categories*, 35:1576–1607, 2020.
- 24 D. Trotta and M.E. Maietti. Generalized existential completions, regular and exact completions. *Preprint*, 2021.

Abstract Congruence Criteria for Weak Bisimilarity

Stelios Tsampas ✉ 

KU Leuven, Belgium

Christian Williams ✉ 

University of California, Riverside, CA, USA

Andreas Nuyts ✉ 

Vrije Universiteit Brussel, Belgium

Dominique Devriese ✉ 

Vrije Universiteit Brussel, Belgium

Frank Piessens ✉ 

KU Leuven, Belgium

Abstract

We introduce three general compositionality criteria over operational semantics and prove that, when all three are satisfied together, they guarantee weak bisimulation being a congruence. Our work is founded upon Turi and Plotkin’s mathematical operational semantics and the coalgebraic approach to weak bisimulation by Brengos. We demonstrate each criterion with various examples of success and failure and establish a formal connection with the simply WB cool rule format of Bloom and van Glabbeek. In addition, we show that the three criteria induce lax models in the sense of Bonchi et al.

2012 ACM Subject Classification Theory of computation → Categorical semantics

Keywords and phrases Structural Operational Semantics, distributive laws, weak bisimilarity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.88

Acknowledgements This research was partially funded by the Research Fund KU Leuven. Andreas Nuyts was supported by a grant of the Research Foundation – Flanders (FWO).

1 Introduction

The problem of *full abstraction* for programming language semantics [21, 13], i.e. the perfect agreement between a denotational and an operational specification, has been both significant and enduring. It requires that the denotational semantics, which bestows each program with a denotation, a meaning, is sufficiently coarse that it does not distinguish terms behaving the same operationally. At the same time, the denotational semantics must remain a *congruence* [36], to make the semantics *compositional*: the denotation of a composite term is fully determined by the denotations of its subterms irrespective of their internal structure.

From an operational point of view, the choice of behavioral equivalence is generally open. Bisimilarity, trace equivalence, weak bisimilarity etc. are all potentially applicable. However, bisimilarity is often too strong for practical purposes. For instance, labelled transition systems [39] (LTS) may perform invisible steps which are not ignored by bisimilarity as opposed to *weak* bisimilarity [18, 19]. This is taken a step further in programming language semantics, where a natural definition of program equivalence is that of the *largest adequate* (w.r.t. observing termination) *congruence* relation [25]. This relation is also known as *contextual equivalence*, the crown jewel of program equivalences, and can be reformulated in a more explicit manner as *Morris-style* contextual equivalence [20, 13, 24], i.e. indistinguishability under all program contexts.



© Stelios Tsampas, Christian Williams, Andreas Nuyts, Dominique Devriese, and Frank Piessens; licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 88; pp. 88:1–88:23

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Despite being the subject of vigorous research, proving that coarser behavioral equivalences are congruences remains a hard problem. Weak bisimilarity in particular stands out as a popular equivalence that has seen widespread usage in the literature yet has been proven hard to reason with [30, 31, 3]. To that end, various powerful methods exist for proving congruence-closedness of bisimilarity, like Howe’s method [15, 16] and *logical relations* [10, 22, 23], yet the machinery involved is complicated and non-trivial to execute correctly. In addition, there have been the so-called *cool congruence formats* for weak bisimulation introduced by Bloom [4] and van Glabbeek [37] that ensure weak bisimilarity being a congruence, but only if the semantics adhere to the rule formats.

In this work, we propose three general *compositionality criteria* over operational specifications that, when all three are satisfied, guarantee weak bisimilarity being a congruence. The foundation of our approach is categorical: on the one hand, the framework of *mathematical operational semantics* introduced by Turi and Plotkin [35] acts as an ideal abstract setting to explore programming language semantics. On the other hand, the coalgebraic approach introduced by Brengos [8, 9] describes a seamless way to define weak bisimulation in categories of coalgebras, should the underlying behavior be monadic and equipped with an order structure. With that in mind, the three criteria, which we name *continuity*, *unitality* and *observability*, essentially characterize how the semantics interact with the order structure of the behavior.

Related work

As mentioned earlier, Bloom and van Glabbeek have introduced the cool congruence formats; weak bisimulation for any system given in these formats is guaranteed to be a congruence. In this work we are also able to establish a formal connection with the *simply WB cool* format for LTSs (Theorem 3.16), specifically that any system given in the simply WB cool rule format automatically satisfies the three criteria. In that sense, our three criteria form a broader, less restrictive approach on the same problem (extended beyond LTSs). Furthermore, the many examples provided by van Glabbeek [37] help explain this connection and at the same time act as excellent hands-on case studies for our three criteria.

Apart from the aforementioned work of Brengos, various approaches at the hard problem of coalgebraic weak bisimulation have been proposed [27, 29, 12, 5]. Of particular interest is the work of Bonchi et al. [6] on up-to techniques [7] for weak bisimulations in the context of mathematical operational semantics. The main theoretical device in their work is that of a *lax model*, a relaxation of the notion of a *bialgebra*. We relate lax models with our work by showing that specifications satisfying the three criteria induce lax models (Theorem 3.21) and argue that, as a formal method, our criteria are significantly easier to prove.

Paper outline

In Section 2.1 we introduce Turi and Plotkin’s mathematical operational semantics [35] as well as the two “running” example systems used throughout the paper. In Section 2.2 we present the work of Brengos on weak bisimulation [8, 9] and show how it applies to our two examples. We expand on our contribution in Section 3, where we introduce the three criteria and put them to the test against our main examples as well as examples from van Glabbeek [37]. We subsequently formalize the connection of our three criteria with the simply WB cool rule format (Theorem 3.16) and then move on to present our main theorem (Theorem 3.18). Finally, in Section 3.3 we show how our three criteria induce lax models in the sense of Bonchi et al. [6].

2 Preliminaries

2.1 Mathematical Operational Semantics

We first summarize the basic framework of Turi and Plotkin's *mathematical operational semantics* [35]. The idea is that operational semantics correspond to *distributive laws* of varying complexity on a base category \mathbb{C} . For our work we need only consider the most important form of distributive laws, that of *GSOS laws* [35], as they are in an 1-1 correspondence with the historically significant *GSOS rule format* [26].

► **Definition 2.1.** *Let endofunctors $\Sigma, B : \mathbb{C} \rightarrow \mathbb{C}$ on a cartesian category \mathbb{C} . A GSOS law of Σ over B is a natural transformation $\rho : \Sigma(\text{Id} \times B) \Rightarrow B\Sigma^*$, where Σ^* is the free monad over Σ .*

Endofunctors Σ and B are understood as the syntax and behavior (resp.) of a system whereas ρ represents the semantics. Over the course of the paper we will also be using an alternative representation of GSOS laws given by the following correspondence.

► **Proposition 2.2.** *GSOS laws $\rho : \Sigma(\text{Id} \times B) \Rightarrow B\Sigma^*$ are equivalent to natural transformations $\lambda : \Sigma^*(\text{Id} \times B) \Rightarrow B\Sigma^*$ respecting the structure of Σ^* as follows:*

$$\begin{array}{ccc}
 \Sigma^*\Sigma^*(\text{Id} \times B) & \xrightarrow{\Sigma^*(\Sigma^*\pi_1, \lambda)} & \Sigma^*(\text{Id} \times B)\Sigma^* & \xrightarrow{\lambda_{\Sigma^*}} & B\Sigma^*\Sigma^* & & \text{Id} \times B & & \\
 \downarrow \mu_{(\text{Id} \times B)} & & & & \downarrow B\mu & & \eta_{(\text{Id} \times B)} \downarrow & \searrow B\eta \circ \pi_2 & \\
 \Sigma^*(\text{Id} \times B) & \xrightarrow{\lambda} & B\Sigma^* & & \Sigma^*(\text{Id} \times B) & \xrightarrow{\lambda} & B\Sigma^* & &
 \end{array}$$

► **Remark 2.3.** We shall be calling both GSOS laws ρ and natural transformations λ , as used in Proposition 2.2, *GSOS laws* for the sake of brevity.

GSOS laws induce *bialgebras*, i.e. algebra-coalgebra pairs that agree with the semantics.

► **Definition 2.4.** *A bialgebra for a GSOS law $\lambda : \Sigma^*(\text{Id} \times B) \Rightarrow B\Sigma^*$ (resp. $\rho : \Sigma(\text{Id} \times B) \Rightarrow B\Sigma^*$) is a Σ -algebra, B -coalgebra pair $\Sigma X \xrightarrow{g} X \xrightarrow{h} BX$ that commutes with λ (ρ):*

$$\begin{array}{ccc}
 \Sigma^* X \xrightarrow{g^\#} X \xrightarrow{h} BX & & \Sigma X \xrightarrow{g} X \xrightarrow{h} BX \\
 \downarrow \Sigma^*(1, h) & \uparrow Bg^\# & \downarrow \Sigma(1, h) & \uparrow Bg^\# \\
 \Sigma^*(X \times BX) \xrightarrow{\lambda} B\Sigma^* X & & \Sigma(X \times BX) \xrightarrow{\rho} B\Sigma^* X
 \end{array}$$

Where $g^\#$ is the \mathcal{EM} -algebra induced by g . A bialgebra morphism from $\Sigma X \xrightarrow{g} X \xrightarrow{h} BX$ to $\Sigma Y \xrightarrow{j} Y \xrightarrow{k} BY$ is a map $f : X \rightarrow Y$ that is both a Σ -algebra and a B -coalgebra morphism.

If $a : \Sigma A \xrightarrow{\cong} A$ is the initial Σ -algebra, the algebra of terms, and $z : Z \xrightarrow{\cong} BZ$ is the final B -coalgebra, the coalgebra of behaviors, the following proposition promotes a GSOS law to an *operational semantics* of a language, in the form of a morphism f mapping programs living in A to behaviors in Z .

► **Proposition 2.5** (From [17, 35]). *Let endofunctors $\Sigma, B : \mathbb{C} \rightarrow \mathbb{C}$ on a cartesian category \mathbb{C} such that $a : \Sigma A \xrightarrow{\cong} A$ is the initial Σ -algebra and $z : Z \xrightarrow{\cong} BZ$ is the final B -coalgebra. Every GSOS law $\lambda : \Sigma^*(\text{Id} \times B) \Rightarrow B\Sigma^*$ induces a unique initial λ -bialgebra $\Sigma A \xrightarrow{a} A \xrightarrow{h} BA$, the operational model of the language, and a unique final λ -bialgebra $\Sigma Z \xrightarrow{g} Z \xrightarrow{z} BZ$, the denotational model. In addition, there exists a unique λ -bialgebra morphism $f : A \rightarrow Z$, mapping every program in A to its behavior in Z .*

The fact that $\text{map } f : A \rightarrow Z$ is an algebra homomorphism is a fundamental well-behavedness property of GSOS laws, as it implies that bisimilarity, defined as equality under f , is a *congruence*, i.e. is respected by all syntactic operators.

The categorical interpretation of GSOS rules can be better understood by the following examples, which serve as the “running” examples of this paper.

► **Example 2.6** (A *While* language). We introduce *While*, a basic imperative language with a mutable state, whose syntax is generated by the following grammar:

$$\langle \text{prog} \rangle ::= \text{skip} \mid v := \langle \text{expr} \rangle \mid \langle \text{prog} \rangle ; \langle \text{prog} \rangle \mid \text{while } \langle \text{expr} \rangle \langle \text{prog} \rangle$$

The statements are the standard **skip**, assignment $:=$, sequential composition $;$ and **while**-loops. Expressions and assignments act on a variable store whose type we denote as S and we shall be writing $[e]_s$ to denote evaluation of an expression e under variable store s .

The syntax of *While* corresponds to the **Set**-endofunctor $\Sigma \triangleq \top \uplus (V \times \text{Exp}) \uplus \text{Id}^2 \uplus (\text{Exp} \times \text{Id})$, with the set of *While*-programs A as the carrier of the initial Σ -algebra $a : \Sigma A \xrightarrow{\cong} A$. The typical behavior functor for deterministic systems with mutable state is $[S \times (\{\checkmark\} \uplus \text{Id})]^S$, where $\{\checkmark\} \cong \top$ is populated by the element that denotes *termination*, but we will instead use $T \triangleq [\mathcal{P}_c(S \times (\{\checkmark\} \uplus \text{Id}))]^S$, where \mathcal{P}_c is the *countable* power-set monad. This way the behavior can be equipped with both a monadic and an order structure, given by inclusion (see Section 2.2), while also allowing for non-determinism. The carrier of the final coalgebra $z : Z \xrightarrow{\cong} TZ$ is the set of behaviors acting on variable stores S returning countably many new stores and possibly new behaviors.

$$\begin{array}{ccc} \text{skip} \frac{}{s, \text{skip} \rightarrow s, \checkmark} & \text{asn} \frac{}{s, v := e \rightarrow s_{[v \leftarrow [e]_s]}, \checkmark} & \text{seq1} \frac{s, p \rightarrow s', \checkmark}{s, p; q \rightarrow s', q} \\ \text{seq2} \frac{s, p \rightarrow s', p'}{s, p; q \rightarrow s', p'; q} & \text{w1} \frac{[e]_s = 0}{s, \text{while } e \text{ } p \rightarrow s, \checkmark} & \text{w2} \frac{[e]_s \neq 0 \quad q \triangleq \text{while } e \text{ } p}{s, q \rightarrow s, p; q} \end{array}$$

The above semantics determines a GSOS law $\rho : \Sigma(\text{Id} \times T) \Longrightarrow T\Sigma^*$ in the category **Set** of sets and (total) functions, or equivalently a natural transformation $\lambda : \Sigma^*(\text{Id} \times T) \Longrightarrow T\Sigma^*$. This example is covered in the literature [34, 33], but we include the definition for posterity.

► **Definition 2.7** (GSOS law of *While*).

$$\begin{array}{ll} \rho_X : \Sigma(X \times TX) & \rightarrow T\Sigma^* X \\ (x, f) ; (y, g) & \mapsto \lambda s. (\{(s', y) \mid (s', \checkmark) \in f(s)\} \cup \{(s', (x' ; y)) \mid (s', x') \in f(s)\}) \\ \text{while } e (x, f) & \mapsto \lambda s. \begin{cases} \{(s, (x ; \text{while } e \text{ } x))\} & \text{if } [e]_s \neq 0 \\ \{(s, \checkmark)\} & \text{if } [e]_s = 0 \end{cases} \\ \text{skip} & \mapsto \lambda s. \{(s, \checkmark)\} \\ v := e & \mapsto \lambda s. \{(s_{[v \leftarrow ev]}, \checkmark)\} \text{ for } ev = [e]_s \end{array}$$

To see how the semantics is connected to the GSOS law, consider rules seq1 and seq2, corresponding to the first line in ρ . Roughly, writing $s, p \rightarrow s', \checkmark$ denotes that $(s', \checkmark) \in f(s)$. The fact that $s, p \rightarrow s', \checkmark$ and $s, p \rightarrow s', q$ are rule premises is reflected in the construction of the set of transitions. Note also that q is used in the conclusion of rules seq1 and seq2. Similarly in ρ we can see that y is present in both the left and right side of ρ , which is why the shape of the behavior of subterms is $X \times TX$ rather than simply TX .

► **Example 2.8** (A simple process calculus). We introduce a simple process calculus, or *SPC* for short, based on the classic *Calculus of communicating systems* introduced by Milner [18]. Its syntax is generated by the following grammar:

$$\langle p \rangle ::= 0 \mid \delta. \langle p \rangle \mid \langle p \rangle \parallel \langle p \rangle \mid \langle p \rangle + \langle p \rangle$$

From left to right we have the *null process* 0, *prefixing* of a process p with action $\delta \in \Delta_\tau = \Delta \cup \overline{\Delta} \cup \{\tau\}$ living in a set composed of actions Δ , coactions $\overline{\Delta}$ and an *internal* action τ , *parallel composition* and finally *non-deterministic choice*.

$$\begin{array}{ccc} \text{sum1} \frac{P \xrightarrow{\delta} P'}{P + Q \xrightarrow{\delta} P'} & \text{sum2} \frac{Q \xrightarrow{\delta} Q'}{P + Q \xrightarrow{\delta} Q'} & \text{com1} \frac{P \xrightarrow{\delta} P'}{P \parallel Q \xrightarrow{\delta} P' \parallel Q} \\ \text{com2} \frac{Q \xrightarrow{\delta} Q'}{P \parallel Q \xrightarrow{\delta} P \parallel Q'} & \text{syn} \frac{\alpha \in \Delta \quad P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\overline{\alpha}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} & \text{prefix} \frac{}{\delta.P \xrightarrow{\delta} P} \end{array}$$

Similarly to Example 2.6, the semantics forms a GSOS law $\rho : \Sigma(\text{Id} \times T) \Longrightarrow T\Sigma^*$ in **Set** for $\Sigma \triangleq \top \uplus (\Delta_\tau \times \text{Id}) \uplus \text{Id}^2 \uplus \text{Id}^2$ and $T \triangleq \mathcal{P}_c(\Delta_\tau \times \text{Id})$. In this case, the carrier of the final coalgebra $z : Z \xrightarrow{\cong} TZ$ is the set of all strongly extensional (meaning that distinct children of nodes are not bisimilar), countably-branching, Δ_τ -labelled trees [2, §4].

► **Definition 2.9** (GSOS law of *SPC*).

$$\begin{array}{ll} \rho_X : \Sigma(X \times TX) & \mapsto \quad T\Sigma^* X \\ \delta.(x, W) & \mapsto \quad \{(\delta, x)\} \\ (x, X) \parallel (y, Z) & \mapsto \quad \{(\delta, (x' \parallel y)) \mid (\delta, x') \in W\} \cup \{(\delta, (x \parallel y')) \mid (\delta, y') \in Z\} \cup \\ & \quad \{(\tau, (x' \parallel y')) \mid (\alpha, x') \in W \wedge (\overline{\alpha}, y') \in Z\} \\ (x, W) + (y, Z) & \mapsto \quad W \cup Z \end{array}$$

2.2 Order-enrichment

Order-enriched categories [38] typically equip their hom-sets with an *order* structure. While there are many forms of order-enrichment, one “nice” such form that is convenient for our purposes is $\omega\text{-Cpo}_{id}^\vee$ -enrichment.

► **Definition 2.10** ([9, §2.3]). *A category \mathbb{C} is $\omega\text{-Cpo}_{id}^\vee$ -enriched when*

- *Every hom-set $\mathbb{C}(X, Y)$ carries a partial order \leq and has all finite joins \vee .*
- *Composition is left-distributive over finite joins, i.e. given any morphisms f, g, i with suitable domains and codomains, $i \circ (f \vee g) = i \circ f \vee i \circ g$.*
- *Every ascending ω -chain $f_0 \leq f_1 \leq \dots$ for $f_i \in \mathbb{C}(X, Y)$ has a supremum $\bigvee_i f_i \in \mathbb{C}(X, Y)$.*
- *Composition $- \circ - : \mathbb{C}(X, Y) \times \mathbb{C}(Y, Z) \rightarrow \mathbb{C}(X, Z)$ is continuous, meaning that for any ascending ω -chains f_i, g_i and morphisms f, g with suitable (co)domains, we have*

$$g \circ \bigvee_i f_i = \bigvee_i (g \circ f_i) \quad \text{and} \quad \left(\bigvee_i g_i \right) \circ f = \bigvee_i (g_i \circ f)$$

For reasons that will become clear in Section 2.3, we are interested in monads whose Kleisli category is $\omega\text{-Cpo}_{id}^\vee$ -enriched as we are looking to use them as behaviors in our distributive laws and exploit their order structure. Examples of such monads are the powerset \mathcal{P} [9, §4.1], the countable powerset \mathcal{P}_c [9, §4.3] and the convex combination monad \mathcal{CM} [9, §4.3].

► **Example 2.11** (Continuation of Example 2.6). The monad structure $\langle T, \eta, \mu \rangle$ for $T \triangleq [\mathcal{P}_c(S \times (\{\checkmark\} \uplus \text{Id}))]^S$ is given by $\eta(x)(s) = \{(s, x)\}$ and $\mu(f)(s) = \bigcup_{(s', g) \in f(s)} \begin{cases} (s', \checkmark) & \text{if } g = \checkmark \\ g(s') & \text{if } g \neq \checkmark \end{cases}$.

The $\omega\text{-Cpo}_{id}^\vee$ -enrichment of $\mathcal{Kl}(T)$ stems from the fact that the countable powerset monad \mathcal{P}_c is itself $\omega\text{-Cpo}_{id}^\vee$ -enriched¹. More specifically, we have $f \leq g \iff \forall x, s. f(x)(s) \subseteq g(x)(s)$, $f \vee g \triangleq \lambda x. \lambda s. [f(x)(s) \cup g(x)(s)]$ and $\bigvee_i f_i = \lambda x. \lambda s. \bigcup_i f_i(x)(s)$.

► **Example 2.12** (Continuation of Example 2.8). The monad structure $\langle T, \eta, \mu \rangle$ for $T \triangleq \mathcal{P}_c(\Delta_\tau \times \text{Id})$ was developed by Brengos [8, §4.1] as a central, demonstrative example of the role of monads in coalgebraic weak bisimulation. The unit is simply $\eta(x) = \{(\tau, x)\}$ and the join $\mu_X : \mathcal{P}_c[\Delta_\tau \times \mathcal{P}_c(\Delta_\tau \times X)] \rightarrow \mathcal{P}_c(\Delta_\tau \times X)$ is $\mu = \mu_{\mathcal{P}_c} \circ \mathcal{P}_c \mu_\Delta \circ \mu_{\mathcal{P}_c} \circ \mathcal{P}_c st$ where $st_{X,Y} : X \times \mathcal{P}_c Y \rightarrow \mathcal{P}_c(X \times Y)$ is the *tensorial strength* of \mathcal{P}_c given by

$$st_{X,Y}(x, Y) = \{(x, y) \mid y \in Y\}$$

and $\mu_\Delta : \Delta_\tau \times \Delta_\tau \times X \rightarrow \mathcal{P}_c \Delta_\tau X$ is

$$\mu_\Delta = \begin{cases} (\delta, \tau, x) \mapsto \{(\delta, x)\} \\ (\tau, \delta, x) \mapsto \{(\delta, x)\} \\ (\delta_1, \delta_2, x) \mapsto \emptyset \quad \text{when } \delta_1, \delta_2 \neq \tau. \end{cases}$$

In other words, μ will disallow any two-step transition that outputs two visible labels in a row and allow everything else, but only after redundant invisible labels are removed. The motivation behind the definition of μ will become clear in Section 2.3. As for the $\omega\text{-Cpo}_{id}^\vee$ -enrichment of $\mathcal{Kl}(T)$, it is also a consequence of $\mathcal{Kl}(\mathcal{P}_c)$ being $\omega\text{-Cpo}_{id}^\vee$ -enriched, with $f \leq g \iff f(x) \subseteq g(x)$, $f \vee g \triangleq \lambda x. [f(x) \cup g(x)]$ and $\bigvee_i f_i = \lambda x. \bigcup_i f_i(x)$.

2.3 Free monads in $\omega\text{-Cpo}_{id}^\vee$ -enriched categories

We now turn our attention to the coalgebraic approach to weak bisimulation of Brengos [8, 9]. The main idea is that given an endomorphism $\alpha : X \rightarrow X$ in an $\omega\text{-Cpo}_{id}^\vee$ -enriched category, there exists the *free monad* over α , α^* . This process, which we call the *reflexive transitive closure* of α or simply the *rt-closure* of α , can be used to derive saturated transition systems or the *multi-step* evaluation relation of a programming language.

► **Remark 2.13.** In general, a *monad* in a bicategory K is an endomorphism $\epsilon : X \rightarrow X$ equipped with 2-cells $\eta : 1_X \rightarrow \epsilon$ and $\mu : \epsilon \circ \epsilon \rightarrow \epsilon$ subject to the conditions $\mu \circ \epsilon \eta = \mu \circ \eta \epsilon = 1_\epsilon$ and $\mu \circ \epsilon \mu = \mu \circ \mu \epsilon$. One can recover the classic definition of a monad by taking the strict 2-category **Cat** of categories, functors and natural transformation as K . In order-enriched categories, where there is at most one 2-cell between morphisms, usually denoted by \leq , monads are simply endomorphisms $\epsilon : X \rightarrow X$ satisfying $1_X \leq \epsilon$ and $\epsilon \circ \epsilon \leq \epsilon$. Monads in order-enriched categories are known as *closure operators*.

► **Definition 2.14** ([9, Definition 3.6]). *An order-enriched category \mathbb{K} admits free monads if for any endomorphism $\alpha : X \rightarrow X$ there exists a monad α^* such that*

- $\alpha \leq \alpha^*$
- if $\alpha \leq \beta$ for a monad $\beta : X \rightarrow X$ then $\alpha^* \leq \beta$.

The free monad of an endomorphism in $\omega\text{-Cpo}_{id}^\vee$ -enriched categories is the least solution of a certain assignment or, equivalently, the supremum of an ascending ω -chain.

► **Proposition 2.15** ([9, §3.2]). *For an $\omega\text{-Cpo}_{id}^\vee$ -enriched category \mathbb{C} , the free monad $(-)^* : \forall X. \mathbb{C}(X, X) \rightarrow \mathbb{C}(X, X)$ of $\alpha : X \rightarrow X$ is given by $\alpha^* \triangleq \mu x. 1 \vee x \circ \alpha = \bigvee_{n < \omega} (1 \vee \alpha)^n$.*

¹ We (slightly abusively) say that a monad T is $\omega\text{-Cpo}_{id}^\vee$ -enriched whenever $\mathcal{Kl}(T)$ is.

Free monads in $\omega\text{-Cpo}_{id}^{\vee}$ -enriched categories enjoy a number of interesting properties.

► **Lemma 2.16** (Properties of free monads). *For all $\alpha : X \rightarrow X$, $\beta : Y \rightarrow Y$, we have*

- i. $\forall f : X \rightarrow Y. f \circ \alpha \leq \beta \circ f \implies f \circ \alpha^* \leq \beta^* \circ f$
- ii. $\alpha^{**} = \alpha^*$
- iii. $\alpha^* = \alpha^* \circ \alpha^*$
- iv. $1^* = 1$.

We can now revisit Example 2.6 and Example 2.8 to witness how the rt-closure acts on each operational model $h : A \rightarrow TA$. Note that throughout the paper we shall be using the notation $- \diamond -$ to denote composition in Kleisli categories.

► **Example 2.17** (Continuation of Example 2.6). The rt-closure (free monad) h^* of the operational model $h : A \rightarrow TA$ amounts to the reflexive, transitive closure of h . The initial stage $(1 \vee h)^0 = 1_{\mathcal{Kl}(T)} = \eta_A$ takes care of the *reflexive* step, while stages $(1 \vee h)^{n+1} = (1 \vee h)^n \diamond (1 \vee h) = (1 \vee h)^n \vee ((1 \vee h)^n \diamond h)$ amount to the inductive, *transitive* step, which acts according to the definition of monadic composition. For instance, $(s, \text{skip} ; \text{skip})$ weakly transitions to $(s, \text{skip} ; \text{skip})$, (s, skip) and (s, \checkmark) .

► **Example 2.18** (Continuation of Example 2.8). The rt-closure h^* amounts to the saturation of $h : A \rightarrow TA$. The unit η establishes $p \xrightarrow{\tau} p$ as a silent step in h^* for $p \in A$ whereas μ ensures that one-step transitions in the saturated system *cannot* produce more than one visible label, i.e. they will always be of the sort of $p \xrightarrow{\tau^*} q \xrightarrow{\delta} r \xrightarrow{\tau^*} s$ or $p \xrightarrow{\tau^*} q$. The reader may refer to [8, 9] for more details.

2.4 Weak bisimulation

We are now ready to define *weak bisimulation* as a special case of an *Aczel-Mendler bisimulation* [1, 32].

► **Definition 2.19.** *Let $f : X \rightarrow FX$ be a coalgebra for a functor $F : \mathbb{C} \rightarrow \mathbb{C}$. An Aczel-Mendler bisimulation, or simply bisimulation, for f is a relation $(\text{span}) X \xleftarrow{r_1} R \xrightarrow{r_2} X$ which is the carrier of a coalgebra $e : R \rightarrow FR$ that lifts to a span of coalgebra homomorphisms, i.e. making the following diagram commute.*

$$\begin{array}{ccccc} FX & \xleftarrow{Fr_1} & FR & \xrightarrow{Fr_2} & FX \\ f \uparrow & & e \uparrow & & f \uparrow \\ X & \xleftarrow{r_1} & R & \xrightarrow{r_2} & X \end{array}$$

In our setting we elect to use the “unoptimized” definition of weak bisimulation as bisimulation on a saturated system, mainly due to its simplicity. Regardless, under the mild condition that *arbitrary cotupling in $\mathcal{Kl}(T)$ is monotonic*, which is true in all of our examples, this definition of weak bisimulation coincides with the traditional one [8, §6].

► **Definition 2.20.** *Assume a coalgebra $h : X \rightarrow TX$ for an $\omega\text{-Cpo}_{id}^{\vee}$ -enriched monad $\langle T, \eta, \mu \rangle$. A weak bisimulation for h is a bisimulation for h^* .*

Two elements $x, y \in X$ are (*weakly*) *bisimilar*, written as $(x \approx y) x \sim y$, if there is a (weak) bisimulation that contains them. If $z : Z \rightarrow TZ$ is the final T -coalgebra, then for every coalgebra $h : X \rightarrow TX$ we write $h^\dagger : X \rightarrow Z$ for the unique coalgebra homomorphism from h to z and h^{\dagger} for the one from h^* to z . The following theorem presents the principle of weak coinduction, i.e. that weakly bisimilar elements are mapped to the same weak behavior.

► **Theorem 2.21** ([8, Theorem 6.8]). *Let $\langle T, \eta, \mu \rangle$ be an $\omega\text{-Cpo}_{\text{id}}^{\vee}$ -enriched monad with final coalgebra $z : Z \rightarrow TZ$. If T preserves weak pullbacks, then the greatest weak bisimulation for a coalgebra $h : X \rightarrow TX$ exists and coincides with the pullback of the equality span $\langle 1_Z, 1_Z \rangle : Z \rightrightarrows Z \times Z$ along $h^\dagger \times h^\dagger : X \times X \rightarrow Z \times Z$.*

The \dagger construction can be applied to any coalgebra, including the final coalgebra $z : Z \rightarrow TZ$. The following lemma shows that h^\dagger can also be recovered via z^\dagger and h^\dagger , a fact that will turn out to be useful in Section 3.

► **Lemma 2.22** ([8, Lemma 6.9]). *For any $h : X \rightarrow TX$, we have $h^\dagger = z^\dagger \circ h^\dagger$.*

3 Weak bisimulation congruence semantics

The theory introduced in Section 2.2 sets the stage for our three compositionality criteria, which ensure that weak bisimilarity is a congruence. In addition, we shall test the criteria on *While* and *SPC* and establish a correspondence with the simply WB cool rule format of van Glabbeek (see [37], definition also included in Appendix A), both formally and through examples, as it also guarantees weak bisimilarity being a congruence and comes with less overhead compared to the WB cool format. Finally, we shall briefly touch on the work of Bonchi et al. [6] and show that systems satisfying the three criteria induce lax models.

3.1 The three compositionality criteria

Simply put, the three criteria ensure that semantics interact with the order structure of the behavior functor in a sensible way. They are *abstract* in that they apply to any GSOS law $\lambda : \Sigma^*(\text{Id} \times T) \Longrightarrow T\Sigma^*$ when $\langle T, \eta, \mu \rangle$ is an $\omega\text{-Cpo}_{\text{id}}^{\vee}$ -enriched monad. As such, we assume the existence of such a T and λ throughout Section 3. We present and explore each criterion individually starting with the simplest of the three.

► **Criterion 1** (Continuity). *For any ascending ω -chain $f_0 \leq f_1 \leq \dots : X \rightarrow TX$ the following condition applies:*

$$\lambda \circ \Sigma^* \langle 1, \bigvee_i f_i \rangle = \bigvee_i \lambda \circ \Sigma^* \langle 1, f_i \rangle \quad (1)$$

Alternatively, we can write the above using $\rho : \Sigma(\text{Id} \times T) \Longrightarrow T\Sigma^*$ as

$$\rho \circ \Sigma \langle 1, \bigvee_i f_i \rangle = \bigvee_i \rho \circ \Sigma \langle 1, f_i \rangle. \quad (2)$$

► **Proposition 3.1.** *Equation (1) and Equation (2) are equivalent.*

We can compare Criterion 1 to the *local continuity* property of lifted functors in Kleisli categories [14, §2.3], i.e. lifted functors respecting the $\omega\text{-Cpo}_{\text{id}}^{\vee}$ -enrichment structure of the category. When it comes to semantics, the following example from van Glabbeek [37, Example 2] underlines what sort of rules may violate Criterion 1.

► **Example 3.2** (Illegal rules). Let us extend *SPC* with a new unary operator, $\llbracket \langle p \rangle \rrbracket$, subject to the following GSOS rules applying to specific actions $\alpha, b \in \Delta$:

$$\text{pos} \frac{P \xrightarrow{\tau} P'}{\llbracket P \rrbracket \xrightarrow{\tau} \llbracket P' \rrbracket} \quad \text{neg} \frac{P \xrightarrow{a} \cdot}{\llbracket P \rrbracket \xrightarrow{b} 0}$$

Rule **neg** allows a process that cannot perform an a -transition to terminate with a b -transition. This rule violates Criterion 1, which can be witnessed by testing the criterion with $(f_0(p) = \{(\tau, p')\}) \leq (f_1(p) = \{(a, q), (\tau, p')\})$, as only f_0 is able to induce a b -transition. In addition, weak bisimulation fails to be a congruence as $\tau.\alpha.0 \approx \alpha.0$ but $\lfloor \tau.\alpha.0 \rfloor \not\approx \lfloor \alpha.0 \rfloor$.

Rule **neg** in the above example includes a negative premise. A GSOS specification that has no negative premises - conclusions are never negative - is called *positive*. Positive GSOS specifications correspond to *monotone* GSOS laws (see [11] and also [6, Equation 7]), in the sense that $g \leq f \implies \lambda \circ \Sigma^* \langle 1, g \rangle \leq \lambda \circ \Sigma^* \langle 1, f \rangle$. This is weaker than continuity (Criterion 1), in that continuous GSOS laws are monotone but not the other way around, but one has to look very hard to find semantics that are monotone yet not continuous.

► **Example 3.3** (A non-continuous monotone rule). We substitute the set of visible actions of *SPC* with \mathbb{N} and define a new operator, $\lfloor _ \rfloor$, subject to rule **mon** $\frac{P \xrightarrow{\infty}}{\lfloor P \rfloor \xrightarrow{\tau} 0}$, where

$P \xrightarrow{\infty}$ denotes that P can perform an infinite number of transitions, i.e. set $\{(\delta, P') \mid P \xrightarrow{\delta} P'\}$ is infinite. Even though this is a monotone rule, it is not continuous. Consider for instance the ascending ω -chain $f_i : X \rightarrow TX$ for some set of processes X with $f_i(x) = \{(0, x), (1, x) \dots (i, x)\}$. Notice that $(\tau, x) \subseteq \lambda \circ \Sigma^* \langle 1, \bigvee_i f_i \rangle$, but $(\tau, x) \not\subseteq \bigvee_i \lambda \circ \Sigma^* \langle 1, f_i \rangle$.

► **Example 3.4** (Continuation of Example 2.6). Back to our *While* language, Criterion 1 is trivially true for all terms except for sequential composition, which is slightly more involved. In this case, we can see that $\rho_X((x, \bigvee_i f_i(x)) ; (y, \bigvee_i f_i(y)))$ is mapped to

$$\begin{aligned} \lambda s. \{ (s', y) \mid (s', \checkmark) \in \bigvee_i f_i(x)(s) \} \cup \{ (s', (x' ; y)) \mid (s', x') \in \bigvee_i f_i(y)(s) \} \\ = \lambda s. \bigvee_i \{ (s', y) \mid (s', \checkmark) \in f_i(x)(s) \} \cup \{ (s', (x' ; y)) \mid (s', x') \in f_i(y)(s) \}, \end{aligned}$$

which is precisely $\bigvee_i (\rho_X((x, f_i(x)) ; (y, f_i(y))))$.

► **Example 3.5** (Continuation of Example 2.8). The transitions of prefix expressions such as δP are, for any given P , independent of (the transitions of) P thus the prefix rule is trivially continuous. Transitions for parallel composition and non-deterministic choice are basically unions of transitions of their subterms and hence satisfy Criterion 1.

► **Criterion 2** (Unitality). For any $f : X \rightarrow TX$,

$$\begin{array}{ccc} \Sigma^* X & \xrightarrow{(\lambda_X \circ \Sigma^* \langle 1, f \rangle)^*} & \\ \Sigma^* \langle 1, \eta_X \vee f \rangle \downarrow & \leq & \\ \Sigma^* (X \times TX) & \xrightarrow{\lambda_X} & T\Sigma^* X \end{array}$$

This criterion characterizes how the semantics deal with internal steps, here represented by the monadic unit η . The right path on the diagram represents the rt-closed (weak) transitions of a composite term, the subterms of which (strongly) transition according to f . On the other hand, the left path represents the strong transitions of a composite term, the subterms of which may also perform internal steps. This criterion, which somewhat resembles the identity condition for lifting functors to Kleisli categories [14, §2.2], dictates that adding arbitrary internal steps to subterms should not lead to extraneous, meaningful observations.

88:10 Abstract Congruence Criteria for Weak Bisimilarity

► **Remark 3.6.** A slightly stronger but simpler formulation of Criterion 2 based on ρ is

$$\begin{array}{ccc} \Sigma X & \xrightarrow{(\rho_X \circ \Sigma\langle 1, f \rangle) \vee (\eta_X \circ \theta_X)} & \\ \Sigma\langle 1, \eta_X \vee f \rangle \downarrow & \leq & \\ \Sigma(X \times TX) & \xrightarrow{\rho_X} & T\Sigma^* X \end{array}$$

where $\theta : \Sigma \Longrightarrow \Sigma^*$ is the universal natural transformation sending Σ to its free monad. Compared to the original criterion, which asks for the transitions induced by internal steps to *eventually* appear on the right side, this version asks for said transition to appear either on *step 0*, the *reflexive*, identity step (hence the added $\eta_X \circ \theta_X$), or *step 1*, i.e. the transitions induced immediately by f . We can apply similar logic to Proposition 3.1 to show that this version of Criterion 2 is stronger.

Criterion 2 works in the same way as the “patience rule” requirement of the simply WB cool rule format [37, Definition 8, item 2], which dictates that the only rules with τ -premises are patience rules. For instance, the patience rule for a unary operator $o(p)$ is $\frac{p \xrightarrow{\tau} p'}{o(p) \xrightarrow{\tau} o(p')}$. It is clear that patience rules are achieving the same effect of forcing composite terms to *only* relay silent steps of subterms.

► **Example 3.7** ([37, Example 4]). We extend SPC with $\lfloor _ \rfloor$ and introduce the following impatient rules:

$$\text{pat} \frac{P \xrightarrow{\tau} P'}{\lfloor P \rfloor \xrightarrow{\tau} \lfloor P' \rfloor} \quad \text{imp} \frac{P \xrightarrow{\tau} P'}{\lfloor P \rfloor \xrightarrow{c} \lfloor P' \rfloor}$$

Rule **imp** violates Criterion 2, as taking $f = \lambda x. \emptyset$ will not induce the c -transition present in the left path. Weak bisimulation fails to be a congruence as $0 \approx \tau.0$ but $\lfloor 0 \rfloor \not\approx \lfloor \tau.0 \rfloor$.

► **Example 3.8** (Continuation of Example 2.6). Language *While* actually respects the stronger version of Criterion 2 found in Remark 3.6. For **skip**, assignment and **while**-loops the transitions induced by $\rho_X \circ \Sigma\langle 1, f \rangle$ are the same regardless of f . This is not the case for sequential composition, but we observe that the left path always leads to $s, x; y \rightarrow s, x; y$, which is covered by the added $\eta_X \circ \theta_X$ on the right path.

► **Example 3.9** (Continuation of Example 2.8). SPC provides for a good example of failure of Criterion 2 as it echoes the well-known fact that weak bisimilarity is not compatible with non-deterministic choice in a manner similar to Example 3.7. In particular, the left path always assigns $x+y$ transitions $\{(\tau, x), (\tau, y)\}$ but for $f = \lambda x. \emptyset$ we have $(\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* = \{(\tau, x+y)\}$. Clearly $\{(\tau, x), (\tau, y)\} \not\subseteq \{(\tau, x+y)\}$ and so Criterion 2 is not satisfied. We can witness the incompatibility of non-deterministic choice by taking a cue from the failing instance and use a process which has no transitions, i.e. the null process: $0 \approx \tau.0$ but $\delta.0 + 0 \not\approx \delta.0 + \tau.0$.

► **Criterion 3** (Observability). For any $f : X \rightarrow TX$,

$$\lambda_X \circ \Sigma^*\langle 1, f \diamond f \rangle \leq (\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \tag{3}$$

Equivalently, we can reformulate the above as

$$\rho_X \circ \Sigma\langle 1, f \diamond f \rangle \leq (\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \circ \theta_X \tag{4}$$

where $\theta : \Sigma \Longrightarrow \Sigma^*$ is the universal natural transformation sending Σ to its free monad.

► **Remark 3.10.** The fact that Equation (3) and Equation (4) are equivalent can be proved in a manner similar to Proposition 3.1.

This criterion is roughly a weakening of the associativity condition for liftings to Kleisli categories [14, §2.2]. We can think of $f \diamond f$ as a two-step transition applied to subterms and $\lambda_X \circ \Sigma^* \langle 1, f \diamond f \rangle$ as the act of a context inspecting zero or more subterms performing that two-step transition. The criterion relates the information obtained by contexts when inspecting two-step transitions as opposed to inspecting each step individually, possibly many times over. Specifically, it ensures that the former always carries *less* information than the latter. A further way to interpret this criterion is that inspecting an effect *now* instead of *later* does not produce new outcomes.

Criterion 3 is more complex to explain in terms of the simply WB cool format, as requirements 1,3,4,5 in Definition 8 from van Glabbeek [37] all contribute towards observations on visible transitions not being affected by silent transitions, regardless of when the latter occur. First, let us look at *straightness*. An operator is straight if it has no rules where a variable occurs multiple times in the left-hand side of its premises. The following example shows how non-straight rules can lead to issues.

► **Example 3.11** ([37, Example 3]). Let operator $\llbracket _ \rrbracket$ subject to the following rules applying to specific $a, b, c \in \Delta$:

$$\text{pat} \frac{P \xrightarrow{\tau} P'}{\llbracket P \rrbracket \xrightarrow{\tau} \llbracket P' \rrbracket} \quad \text{cur} \frac{P \xrightarrow{a} Q \quad P \xrightarrow{b} W}{\llbracket P \rrbracket \xrightarrow{c} \llbracket Q \rrbracket}$$

We can see how rule **cur** violates Criterion 3 by taking $f(p) = \{(a, q), (\tau, w)\}$, $f(q) = \{(\tau, q)\}$ and $f(w) = \{(b, w)\}$. Since $(f \diamond f)(p) = \{(a, q), (b, w)\}$, running $\lambda_X \circ \Sigma^* \langle 1, f \diamond f \rangle$ on $\llbracket p \rrbracket$ induces a c -transition, which does not occur on the right side. With rule **cur** weak bisimilarity is not a congruence, as $\alpha.0 + b.0 + \tau.b.0 \approx \alpha.0 + \tau.b.0$ but $\llbracket \alpha.0 + b.0 + \tau.b.0 \rrbracket \not\approx \llbracket \alpha.0 + \tau.b.0 \rrbracket$.

Requirements 3 and 4 in the definition of the simply WB cool format underline how the *lack* of patience rules can affect observations.

► **Example 3.12** ([37, Example 5]). Consider operator $\llbracket _ \rrbracket$ subject to rule **oba** $\frac{P \xrightarrow{a} P'}{\llbracket P \rrbracket \xrightarrow{\tau} 0}$ applying to a specific action a . Rule **oba** fails Criterion 3 with $f(p) = \{(\tau, q)\}$ and $f(q) = \{(a, w)\}$, making $(f \diamond f)(p) = \{(a, w)\}$. Running $\lambda_X \circ \Sigma^* \langle 1, f \diamond f \rangle$ on $\llbracket p \rrbracket$ induces a τ -transition, which does not occur on the right side. We can see how $\alpha.0 \approx \tau.\alpha.0$ but $\llbracket \alpha.0 \rrbracket \not\approx \llbracket \tau.\alpha.0 \rrbracket$.

The final requirement is that of *smoothness*. A straight operator for an LTS is smooth if it has no rules where a variable occurs both in the target and in the left-hand side of a premise. Non-smooth rules can also cause problems, as evidenced by the following example.

► **Example 3.13** ([37, Example 7]). Let operator $\llbracket _ \rrbracket$ with the following rules:

$$\text{play} \frac{P \xrightarrow{\delta} P'}{\llbracket P \rrbracket \xrightarrow{\delta} \llbracket P' \rrbracket} \quad \text{pause} \frac{P \xrightarrow{\delta} P'}{\llbracket P \rrbracket \xrightarrow{\delta} \llbracket P \rrbracket}$$

Non-smooth rule **pause** also violates Criterion 3. Take $f(p) = \{(\tau, q)\}$ and $f(q) = \{(a, w)\}$, making $(f \diamond f)(p) = \{(a, w)\}$. Running $\lambda_X \circ \Sigma^* \langle 1, f \diamond f \rangle$ on $\llbracket p \rrbracket$ induces a -transition $(a, \llbracket p \rrbracket)$, but the only a -transitions induced on the other side are $(a, \llbracket q \rrbracket)$ and $(a, \llbracket w \rrbracket)$ instead. Weak bisimilarity fails to be a congruence, with $\alpha.0 + \tau.b.0 \approx \alpha.0 + \tau.b.0 + b.0$ but $\llbracket \alpha.0 + \tau.b.0 \rrbracket \not\approx \llbracket \alpha.0 + \tau.b.0 + b.0 \rrbracket$. The difference here is that only $\llbracket \alpha.0 + \tau.b.0 + b.0 \rrbracket$ is able to perform an α -transition after a b -transition.

► **Example 3.14** (Continuation of Example 2.6). The situation for Criterion 3 is less obvious for both `while`-loops and sequential composition, but still trivial for `skip` and assignment statements. Using the simpler ρ -based formulation of Criterion 3, we have that for `while`-loops, the induced transition is not affected by transitions of subterms, hence $\rho_X \circ \Sigma\langle 1, f \diamond f \rangle$ is always included in the first iteration of $(\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \circ \theta$.

Showing that the criterion is satisfied by an expression $x; y$ for any $x, y \in X$ requires case analysis of $(f \diamond f)(x)$ only, as the rule ignores y . The two cases are:

- $(t, z) \in (f \diamond f)(x)(s)$. In this case x did not terminate, but rather went through an intermediate transition t', z' . According to rule `seq2`, the transition produced by $\rho_X \circ \Sigma\langle 1, f \diamond f \rangle$ is $s, x; y \rightarrow t, z; y$. Going over to $(\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \circ \theta$, the first iteration produces $s, x; y \rightarrow t', z'; y$, and in the second we get $t', z'; y \rightarrow t, z; y$, which is the same result.
- $(t, \checkmark) \in (f \diamond f)(x)(s)$. Here, x terminated producing t either immediately $((t, \checkmark) \in f(x)(s))$ or in two steps $((s', x') \in f(x)(s)$ and $(t, \checkmark) \in f(x')(s'))$. In any case, the transition produced by $\rho_X \circ \Sigma\langle 1, f \diamond f \rangle$ is $s, x; y \rightarrow t, y$. Depending on when x terminated, this transition will be “caught” in either the first or the second iteration of $(\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \circ \theta$.

► **Example 3.15** (Continuation of Example 2.8). When instantiated to *SPC*, the criterion essentially asks if the act of “forgetting” invisible steps of subterms, as imposed by the rules of monadic composition in $\mathcal{Kl}(T)$, gives rise to new transitions for a composite term. In most cases, this is evidently true; consider for instance the parallel composition of two terms $P \parallel Q$, for which $P \xrightarrow{\tau} P' \xrightarrow{\delta} P''$, i.e. $(\tau, P') \in f(P)$ and $(\delta, P'') \in f(P')$. Forgetting the invisible step gives $P \xrightarrow{\delta} P''$ $((\delta, P'') \in (f \diamond f)(P))$, so by rule `com1` we have $P \parallel Q \xrightarrow{\delta} P'' \parallel Q$ on the left-hand side. This transition will occur after two iterations on the right-hand side, as in $P \parallel Q \xrightarrow{\tau} P' \parallel Q \xrightarrow{\delta} P'' \parallel Q$.

Rule `syn` is especially interesting, as it showcases the full power of Criterion 3. There are four different cases where `syn` induces a transition $P \parallel Q \xrightarrow{\tau} P'' \parallel Q''$ on the left side, namely

- $P \xrightarrow{\tau} P' \xrightarrow{\alpha} P''$ and $Q \xrightarrow{\tau} Q' \xrightarrow{\bar{\alpha}} Q''$
- $P \xrightarrow{\tau} P' \xrightarrow{\alpha} P''$ and $Q \xrightarrow{\bar{\alpha}} Q' \xrightarrow{\tau} Q''$
- $P \xrightarrow{\alpha} P' \xrightarrow{\tau} P''$ and $Q \xrightarrow{\bar{\alpha}} Q' \xrightarrow{\tau} Q''$
- $P \xrightarrow{\alpha} P' \xrightarrow{\tau} P''$ and $Q \xrightarrow{\tau} Q' \xrightarrow{\bar{\alpha}} Q''$.

The third and fourth case work similarly to the first and second (resp.) and so we focus on the latter. Either way, the right side of Criterion 3 needs three iterations (meaning $(\lambda_X \circ \Sigma^*\langle 1, f \rangle) \diamond (\lambda_X \circ \Sigma^*\langle 1, f \rangle) \diamond (\rho_X \circ \Sigma\langle 1, f \rangle)$) in order to produce transition $P \parallel Q \xrightarrow{\tau} P'' \parallel Q''$. In the first case, each iteration executes (in sequence) rules `com1`, `com2` and `syn` leading to $P \parallel Q \xrightarrow{\tau} P' \parallel Q \xrightarrow{\tau} P' \parallel Q' \xrightarrow{\tau} P'' \parallel Q''$. In the second case the order changes with rules `com1`, `syn` and `com2` inducing $P \parallel Q \xrightarrow{\tau} P' \parallel Q \xrightarrow{\tau} P'' \parallel Q' \xrightarrow{\tau} P'' \parallel Q''$.

Up to this point, the connection of our criteria with the simply WB cool format has remained informal. The following theorem turns this connection to a formal correspondence.

► **Theorem 3.16.** *Any language in the simply WB cool format satisfies the three compositionality criteria.*

The converse is not true, as for example any simple non-smooth rule satisfying the three criteria, such as $[x] \xrightarrow{c} x$, is not simply WB cool. A proof of Theorem 3.16 is provided in Appendix B.2.

3.2 An algebra for a †

We are now ready to move on to the main theorem of our work, namely the existence of a compatible algebra structure for morphism $h^\dagger : A \rightarrow Z$ mapping every term in A to its weak behavior in Z . First, we present the following intermediate result that plays a catalytic role in the main theorem and is noteworthy in its own right.

► **Proposition 3.17.** *Let $\lambda : \Sigma^*(\text{Id} \times T) \Longrightarrow T\Sigma^*$ be a GSOS law of Σ over T with T being an $\omega\text{-Cpo}_{\text{id}}^\vee$ -enriched monad. If λ satisfies the three compositionality criteria, then for any T -coalgebra $f : X \rightarrow TX$ we have $(\lambda \circ \Sigma^*\langle 1, f^* \rangle)^* = (\lambda \circ \Sigma^*\langle 1, f \rangle)^*$.*

Proof. By antisymmetry on the order structure \leq of T . To avoid unnecessary clutter, for the rest of the proof we shall be writing $\lambda \circ \Sigma^*\langle 1, f \rangle$ as $\bar{\Sigma}f$ and the 1 in $\bar{\Sigma}(1 \vee f)$ will stand for η , the identity morphism in $\mathcal{Kl}(T)$.

- Via Criterion 1 and Lemma 2.16 we have $f \leq f^* \implies \bar{\Sigma}f \leq \bar{\Sigma}f^* \implies (\bar{\Sigma}f)^* \leq (\bar{\Sigma}f^*)^*$. In other words, $(\lambda \circ \Sigma^*\langle 1, f \rangle)^* \leq (\lambda \circ \Sigma^*\langle 1, f^* \rangle)^*$.
- We first show that $\bar{\Sigma}(1 \vee f)^n \leq (\bar{\Sigma}f)^*$ for all $n \in \mathbb{N}$ by induction on n . For $n = 0$ and $n = 1$ and by Criterion 1 (as $1 \leq (1 \vee f)$) and Criterion 2,

$$\bar{\Sigma}(1 \vee f)^0 = \bar{\Sigma}1 \leq \bar{\Sigma}(1 \vee f) = \bar{\Sigma}(1 \vee f)^1 \leq (\bar{\Sigma}f)^* \quad (5)$$

We now have to show that $\bar{\Sigma}((1 \vee f)^{n+1} \diamond (1 \vee f)) \leq (\bar{\Sigma}f)^*$ for some n by making use of the inductive hypothesis $\bar{\Sigma}(1 \vee f)^{n+1} \leq (\bar{\Sigma}f)^*$. To that end, we first note that $1 \leq (1 \vee f)$ and thus, due to $\omega\text{-Cpo}_{\text{id}}^\vee$ -enrichment, we have that for all $n \in \mathbb{N}$,

$$1 \vee f \leq (1 \vee f) \diamond (1 \vee f) \leq \dots \leq (1 \vee f)^{n+1} \implies (1 \vee f)^{n+1} \diamond (1 \vee f) \leq (1 \vee f)^{n+1} \diamond (1 \vee f)^{n+1} \quad (6)$$

Next, by (6), Criterion 1 and Criterion 3,

$$\bar{\Sigma}((1 \vee f)^{n+1} \diamond (1 \vee f)) \leq \bar{\Sigma}((1 \vee f)^{n+1} \diamond (1 \vee f)^{n+1}) \leq (\bar{\Sigma}(1 \vee f)^{n+1})^* \quad (7)$$

The induction hypothesis gives $\bar{\Sigma}(1 \vee f)^{n+1} \leq (\bar{\Sigma}f)^*$ and so (7) becomes

$$\bar{\Sigma}((1 \vee f)^{n+1} \diamond (1 \vee f)) \leq (\bar{\Sigma}(1 \vee f)^{n+1})^* \leq (\bar{\Sigma}f)^{**} = (\bar{\Sigma}f)^* \quad (8)$$

Inequalities (8) and (5) complete the inductive proof that $\forall n \in \mathbb{N}. \bar{\Sigma}(1 \vee f)^n \leq (\bar{\Sigma}f)^*$. Finally, by Proposition 2.15 and Criterion 1:

$$\bar{\Sigma}f^* = \bar{\Sigma} \bigvee_{n < \omega} (1 \vee f)^n = \bigvee_{n < \omega} \bar{\Sigma}(1 \vee f)^n$$

We just proved that every link in the ω -chain is smaller than $(\bar{\Sigma}f)^*$, and thus $\bar{\Sigma}f^* \leq (\bar{\Sigma}f)^*$.

By Definition 2.14, this becomes $(\bar{\Sigma}f^*)^* \leq (\bar{\Sigma}f)^*$, i.e. $(\lambda \circ \Sigma^*\langle 1, f^* \rangle)^* \leq (\lambda \circ \Sigma^*\langle 1, f \rangle)^*$.

Having shown both directions, we end up with $(\lambda \circ \Sigma^*\langle 1, f^* \rangle)^* = (\lambda \circ \Sigma^*\langle 1, f \rangle)^*$. ◀

In other words, the transition system of (arbitrarily deep) contexts with subterms in X is *weakly* equivalent that of contexts having access to all the weak transitions of subterms in X . Proposition 3.17 is what enables the formation of a compatible algebra structure for $h^\dagger : A \rightarrow Z$ by applying it to the final coalgebra $z : Z \xrightarrow{\cong} BZ$. It is currently unclear if the converse of Proposition 3.17 holds.

► **Theorem 3.18 (Main theorem).** *Let $\lambda : \Sigma^*(\text{Id} \times T) \Longrightarrow T\Sigma^*$ be a GSOS law of Σ over T with T being an $\omega\text{-Cpo}_{\text{id}}^\vee$ -enriched monad. If λ satisfies the three compositionality criteria, then for $\Sigma A \xrightarrow{a} A \xrightarrow{h} TA$ and $\Sigma Z \xrightarrow{g} Z \xrightarrow{z} TZ$ as in Proposition 2.5, h^\dagger is a Σ -algebra homomorphism from $a : \Sigma A \xrightarrow{\cong} A$ to $z^\dagger \circ g : \Sigma Z \rightarrow Z$.*

88:14 Abstract Congruence Criteria for Weak Bisimilarity

Proof. We first observe that the following diagram commutes due to finality of z (top left and bottom rectangle) and naturality of λ . Note that the horizontal lines are all T -coalgebras and $g^\#$ is the \mathcal{EM} -algebra induced by the denotational model g .

$$\begin{array}{ccccc}
 \Sigma^* Z & \xrightarrow{\Sigma^* \langle 1, z^* \rangle} & \Sigma^* (Z \times TZ) & \xrightarrow{\lambda} & T\Sigma^* Z \\
 \Sigma^*(z^\dagger) \downarrow & & \downarrow \Sigma^*(\text{Id} \times T)(z^\dagger) & & \downarrow T\Sigma^*(z^\dagger) \\
 \Sigma^* Z & \xrightarrow{\Sigma^* \langle 1, z \rangle} & \Sigma^* (Z \times TZ) & \xrightarrow{\lambda} & T\Sigma^* Z \\
 g^\# \downarrow & & & & \downarrow Tg^\# \\
 Z & \xrightarrow{\cong} & Z & \xrightarrow{z} & TZ
 \end{array}$$

Since rt-closing preserves T -coalgebra homomorphisms, rt-closing the T -coalgebras and finality of z gives us

$$\begin{array}{ccc}
 \Sigma^* Z & \xrightarrow{(\lambda \circ \Sigma^* \langle 1, z^* \rangle)^*} & T\Sigma^* Z \\
 \Sigma^*(z^\dagger) \downarrow & & \downarrow T\Sigma^*(z^\dagger) \\
 \Sigma^* Z & \xrightarrow{(\lambda \circ \Sigma^* \langle 1, z \rangle)^*} & T\Sigma^* Z \\
 g^\# \downarrow & & \downarrow Tg^\# \\
 Z & \xrightarrow{z^*} & TZ \\
 z^\dagger \downarrow & & \downarrow Tz^\dagger \\
 Z & \xrightarrow{\cong} & TZ
 \end{array}$$

Via Proposition 3.17 we have $(\lambda \circ \Sigma^* \langle 1, z^* \rangle)^* = (\lambda \circ \Sigma^* \langle 1, z \rangle)^*$. Since all homomorphisms on final coalgebras are unique, we see that $z^\dagger \circ g^\# = z^\dagger \circ g^\# \circ \Sigma^*(z^\dagger)$, which in turn makes the following diagram commute:

$$\begin{array}{ccccc}
 \Sigma Z & \xrightarrow{\theta_Z} & \Sigma^* Z & \xrightarrow{g^\#} & Z \\
 \Sigma z^\dagger \downarrow & & \Sigma^* z^\dagger \downarrow & & \downarrow z^\dagger \\
 \Sigma Z & \xrightarrow{\theta_Z} & \Sigma^* Z & \xrightarrow{z^\dagger \circ g^\#} & Z
 \end{array}$$

Where $\theta : \Sigma \implies \Sigma^*$ is the universal natural transformation sending Σ to its free monad. But $g^\# \circ \theta = g$ and so we can conclude

$$z^\dagger \circ g = z^\dagger \circ g \circ \Sigma z^\dagger \tag{9}$$

$$\implies z^\dagger \circ g \circ \Sigma h^\dagger = z^\dagger \circ g \circ \Sigma z^\dagger \circ \Sigma h^\dagger \tag{10}$$

$$\implies z^\dagger \circ h^\dagger \circ a = z^\dagger \circ g \circ \Sigma(z^\dagger \circ h^\dagger) \tag{11}$$

$$\implies h^\dagger \circ a = z^\dagger \circ g \circ \Sigma h^\dagger. \tag{12}$$

Equation (10) gives (11) due to h^\dagger being a bialgebra morphism and finally we have (12) by Lemma 2.22, which finishes the proof. \blacktriangleleft

Weak bisimilarity being a congruence is thus a simple corollary of Theorems 2.21 and 3.18.

► Corollary 3.19. *Let $\lambda : \Sigma^*(\text{Id} \times T) \implies T\Sigma^*$ be a GSOS law of Σ over weak-pullback preserving functor T with T being an $\omega\text{-Cpo}_{id}^\vee$ -enriched monad as in Theorem 3.18. If λ satisfies the three compositionality criteria, weak bisimilarity in λ is a congruence.*

3.3 Lax models for weak bisimulation

Up-to techniques for bisimulation [19, 28] are techniques that simplify reasoning about behavioral equivalences, the main idea being that instead of having to show that two processes are included in a bisimulation relation, one can show that they are included in a different relation which is *sound* with respect to bisimulation. Bonchi et al. [6, Corollary 22] proved that weak bisimulation up-to contextual closure is a compatible (and hence sound) up-to technique for systems specified in the simply WB cool rule format. This result is a corollary of their main theorem [6, Theorem 20], which requires the underlying system to be *positive* and also the saturated transition system to be a *lax model* for the given specification, requirements that are automatically true for systems in the simply WB cool format.

As mentioned in Section 3.1, the abstract equivalent of positivity for GSOS specifications is monotonicity, which is weaker than continuity (Criterion 1). Thus, a GSOS law satisfying the three compositionality criteria is monotone. As for lax models, our behaviors come with an order structure and hence we can give the following definition.

► **Definition 3.20** (Lax models for GSOS laws). *Let $\lambda : \Sigma^*(\text{Id} \times T) \Longrightarrow T\Sigma^*$ be a GSOS law of Σ over T with T being an $\omega\text{-Cpo}_{\text{Id}}^{\vee}$ -enriched monad. A lax λ -model is a Σ -algebra and T -coalgebra pair $\Sigma X \xrightarrow{g} X \xrightarrow{h} TX$ making the following diagram commute laxly:*

$$\begin{array}{ccccc} \Sigma^* X & \xrightarrow{g^\#} & X & \xrightarrow{h} & TX \\ \Sigma^*(1, h) \downarrow & & \vee \! \! \! \vee & & \uparrow Tg^\# \\ \Sigma^*(X \times TX) & \xrightarrow{\lambda} & & \xrightarrow{\lambda} & T\Sigma^* X \end{array}$$

Where $g^\#$ is the respective \mathcal{EM} -algebra induced by g .

In other words, a lax model is a relaxed version of a bialgebra (Definition 2.4), implying that *only*, but not necessarily *all*, weak transitions of a composite term can be deduced from the weak transitions of its subterms. One non-example of a lax model is $\Sigma A \xrightarrow{a} A \xrightarrow{h^*} BA$ of *SPC* from Example 2.8. We can use the same problematic case as Example 3.9: the lower path on the bialgebra diagram for process $\delta.0 + 0$ reveals transition $\delta.0 + 0 \xrightarrow{\tau} 0$, which is not a transition of $\delta.0 + 0$.

For all their nice properties, there is little indication as to which GSOS laws have lax models and why. Thus, in the context of our work, it is sensible to ask if the three congruence criteria are adequate with respect to producing lax models for GSOS laws, with the following theorem asserting that this is indeed the case.

► **Theorem 3.21.** *Let $\lambda : \Sigma^*(\text{Id} \times T) \Longrightarrow T\Sigma^*$ be a GSOS law of Σ over T with T being an $\omega\text{-Cpo}_{\text{Id}}^{\vee}$ -enriched monad. If λ satisfies the three compositionality criteria, then for any λ -bialgebra $\Sigma X \xrightarrow{g} X \xrightarrow{h} TX$, $\Sigma X \xrightarrow{g} X \xrightarrow{h^*} TX$ is a lax λ -model.*

Proof. By Lemma 2.16 and the definition of a bialgebra we have $Tg^\# \circ (\lambda_X \circ \Sigma^*(1, h))^* = h^* \circ g^\#$. Thus, to prove that $\Sigma X \xrightarrow{g} X \xrightarrow{h^*} TX$ is lax λ -model, it suffices to show that $\lambda \circ \Sigma^*(1, h^*) \leq (\lambda \circ \Sigma^*(1, h))^*$. by Proposition 3.17 and Definition 2.14 we indeed have that $\lambda \circ \Sigma^*(1, h^*) \leq (\lambda \circ \Sigma^*(1, h))^* = (\lambda \circ \Sigma^*(1, h))^*$. ◀

It is worth noting that compatibility of the up-to context technique for weak bisimulation entails the congruence property of weak bisimilarity, which means that there is an alternative route to Corollary 3.19 via Proposition 3.17 and [6, Theorem 20]. With that in mind, can lax models work as a formal method for proving congruence of weak bisimilarity like our

three criteria? The problem is that proving laxness of a model (typically the initial, rt-closed model $\Sigma A \xrightarrow{a} A \xrightarrow{h^*} TA$) involves non-trivial reasoning on an rt-closed system that is itself defined inductively on the structure of terms. Conversely, our three criteria are significantly easier to establish as they characterize a GSOS law acting on a single layer of syntax.

4 Conclusion

In this paper we presented three abstract criteria over operational semantics, given in the form of Turi and Plotkin's bialgebraic semantics, that guarantee weak bisimilarity being a congruence. We believe that the criteria gracefully balance between generality and usefulness but, as is often the case with abstract results, this is something hard to assess accurately. What is equally important however is that each of the criteria can be given an intuitive explanation as to the kind of restriction it imposes on the semantics. We hope that these insights can contribute towards a conclusive answer to the general problem of full abstraction: the definition of the best adequate denotational semantics, the underlying equivalence of which coincides with contextual equivalence.

References

- 1 Peter Aczel and Nax Paul Mendler. A final coalgebra theorem. In David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts, and Axel Poigné, editors, *Category Theory and Computer Science, Manchester, UK, September 5–8, 1989, Proceedings*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer, 1989. doi:10.1007/BFb0018361.
- 2 Jirí Adámek, Paul Blain Levy, Stefan Milius, Lawrence S. Moss, and Lurdes Sousa. On final coalgebras of power-set functors and saturated trees – to george janelidze on the occasion of his sixtieth birthday. *Applied Categorical Structures*, 23(4):609–641, 2015. doi:10.1007/s10485-014-9372-9.
- 3 Christel Baier and Holger Hermanns. Weak bisimulation for fully probabilistic processes. In Orna Grumberg, editor, *Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22–25, 1997, Proceedings*, volume 1254 of *Lecture Notes in Computer Science*, pages 119–130. Springer, 1997. doi:10.1007/3-540-63166-6_14.
- 4 Bard Bloom. Structural operational semantics for weak bisimulations. *Theor. Comput. Sci.*, 146(1&2):25–68, 1995. doi:10.1016/0304-3975(94)00152-9.
- 5 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. Coinduction up-to in a fibrational setting. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14–18, 2014*, pages 20:1–20:9. ACM, 2014. doi:10.1145/2603088.2603149.
- 6 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. Lax bialgebras and up-to techniques for weak bisimulations. In Luca Aceto and David de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1–4, 2015*, volume 42 of *LIPICs*, pages 240–253. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPICs.CONCUR.2015.240.
- 7 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. A general account of coinduction up-to. *Acta Inf.*, 54(2):127–190, 2017. doi:10.1007/s00236-016-0271-4.
- 8 Tomasz Brengos. Weak bisimulation for coalgebras over order enriched monads. *Logical Methods in Computer Science*, 11(2), 2015. doi:10.2168/LMCS-11(2:14)2015.
- 9 Tomasz Brengos, Marino Miculan, and Marco Peressotti. Behavioural equivalences for coalgebras with unobservable moves. *J. Log. Algebraic Methods Program.*, 84(6):826–852, 2015. doi:10.1016/j.jlamp.2015.09.002.

- 10 Derek Dreyer, Amal Ahmed, and Lars Birkedal. Logical step-indexed logical relations. *Logical Methods in Computer Science*, 7(2), 2011. doi:10.2168/LMCS-7(2:16)2011.
- 11 Marcelo Fiore and Sam Staton. Positive structural operational semantics and monotone distributive laws. In *CMCS'10 Short Contributions*, 2010.
- 12 Sergey Goncharov and Dirk Pattinson. Coalgebraic weak bisimulation from recursive equations over monads. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 196–207. Springer, 2014. doi:10.1007/978-3-662-43951-7_17.
- 13 Andrew D. Gordon. Bisimilarity as a theory of functional programming. *Theor. Comput. Sci.*, 228(1-2):5–47, 1999. doi:10.1016/S0304-3975(98)00353-3.
- 14 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007. doi:10.2168/LMCS-3(4:11)2007.
- 15 Douglas J. Howe. Equality in lazy computation systems. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, pages 198–203. IEEE Computer Society, 1989. doi:10.1109/LICS.1989.39174.
- 16 Douglas J. Howe. Proving congruence of bisimulation in functional programming languages. *Inf. Comput.*, 124(2):103–112, 1996. doi:10.1006/inco.1996.0008.
- 17 Bartek Klin. Bialgebras for structural operational semantics: An introduction. *Theor. Comput. Sci.*, 412(38):5043–5069, 2011. doi:10.1016/j.tcs.2011.03.023.
- 18 Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980. doi:10.1007/3-540-10235-3.
- 19 Robin Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.
- 20 James H. Morris. *Lambda-Calculus Models of Programming Languages*. PhD thesis, Massachusetts Institute of Technology, 1968.
- 21 C.-H. L. Ong. *Correspondence between Operational and Denotational Semantics: The Full Abstraction Problem for PCF*, page 269?356. Oxford University Press, Inc., USA, 1995.
- 22 Andrew M. Pitts. Reasoning about local variables with operationally-based logical relations. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 152–163. IEEE Computer Society, 1996. doi:10.1109/LICS.1996.561314.
- 23 Andrew M. Pitts. A note on logical relations between semantics and syntax. *Log. J. IGPL*, 5(4):589–601, 1997. doi:10.1093/jigpal/5.4.589.
- 24 Andrew M. Pitts. Parametric polymorphism and operational equivalence. *Math. Struct. Comput. Sci.*, 10(3):321–359, 2000. URL: <http://journals.cambridge.org/action/displayAbstract?aid=44651>.
- 25 Andrew M. Pitts. Typed operational reasoning. In Benjamin C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7. The MIT Press, 2004.
- 26 Gordon D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.
- 27 Andrei Popescu. Weak bisimilarity coalgebraically. In *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7–10, 2009. Proceedings*, pages 157–172, 2009. doi:10.1007/978-3-642-03741-2_12.
- 28 Damien Pous and Davide Sangiorgi. Enhancements of the bisimulation proof method. In Davide Sangiorgi and Jan J. M. M. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*, volume 52 of *Cambridge tracts in theoretical computer science*, pages 233–289. Cambridge University Press, 2012.
- 29 Jan Rothe and Dragan Masulovic. Towards weak bisimulation for coalgebras. *Electr. Notes Theor. Comput. Sci.*, 68(1):32–46, 2002. doi:10.1016/S1571-0661(04)80499-7.
- 30 Jan J. M. M. Rutten. A note on coinduction and weak bisimilarity for while programs. *ITA*, 33(4/5):393–400, 1999. doi:10.1051/ita:1999125.

- 31 Davide Sangiorgi and Robin Milner. The problem of “weak bisimulation up to”. In Rance Cleaveland, editor, *CONCUR '92, Third International Conference on Concurrency Theory, Stony Brook, NY, USA, August 24–27, 1992, Proceedings*, volume 630 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 1992. doi:10.1007/BFb0084781.
- 32 Sam Staton. Relating coalgebraic notions of bisimulation. *Log. Methods Comput. Sci.*, 7(1), 2011. doi:10.2168/LMCS-7(1:13)2011.
- 33 Stelios Tsampas, Andreas Nuyts, Dominique Devriese, and Frank Piessens. A categorical approach to secure compilation. In Daniela Petrisan and Jurriaan Rot, editors, *Coalgebraic Methods in Computer Science – 15th IFIP WG 1.3 International Workshop, CMCS 2020, Colocated with ETAPS 2020, Dublin, Ireland, April 25-26, 2020, Proceedings*, volume 12094 of *Lecture Notes in Computer Science*, pages 155–179. Springer, 2020. doi:10.1007/978-3-030-57201-3_9.
- 34 Daniele Turi. Categorical modelling of structural operational rules: Case studies. In *Category Theory and Computer Science, 7th International Conference, CTCS '97, Santa Margherita Ligure, Italy, September 4–6, 1997, Proceedings*, pages 127–146, 1997. doi:10.1007/BFb0026985.
- 35 Daniele Turi and Gordon D. Plotkin. Towards a mathematical operational semantics. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 – July 2, 1997*, pages 280–291, 1997. doi:10.1109/LICS.1997.614955.
- 36 Rob J. van Glabbeek. Full abstraction in structural operational semantics (extended abstract). In Maurice Nivat, Charles Rattray, Teodor Rus, and Giuseppe Scollo, editors, *Algebraic Methodology and Software Technology (AMAST '93), Proceedings of the Third International Conference on Methodology and Software Technology, University of Twente, Enschede, The Netherlands, 21–25 June, 1993, Workshops in Computing*, pages 75–82. Springer, 1993.
- 37 Rob J. van Glabbeek. On cool congruence formats for weak bisimulations. *Theor. Comput. Sci.*, 412(28):3283–3302, 2011. doi:10.1016/j.tcs.2011.02.036.
- 38 Mitchell Wand. Fixed-point constructions in order-enriched categories. *Theor. Comput. Sci.*, 8:13–30, 1979. doi:10.1016/0304-3975(79)90053-7.
- 39 Glynn Winskel and Mogens Nielsen. Models for concurrency. *DAIMI Report Series*, 22(463), November 1993. doi:10.7146/dpb.v22i463.6936.

A The simply WB cool rule format

In this section we introduce a few notions relevant to the simply WB cool rule format taken from [37]. For some n -ary operator o in a language \mathcal{L} , the *rules of o* are all the rules with source $o(x_1, \dots, x_n)$. The following characterize GSOS rules based on the shape(s) of its premise(s) and the shape of its conclusion.

- An operator in \mathcal{L} is *straight* if it has no rules in which a variable occurs multiple times in the left-hand side of its premises.
- An operator is *smooth* if it is straight and has no rules in which a variable occurs both in the target and in the left-hand side of a premise.
- An argument $i \in \mathbb{N}$ of an operator o is *active* if there is a rule of o in which x_i is the left-hand side of a premise.
- A variable x occurring in a term t is *receiving in t* if t is the target of a rule in \mathcal{L} in which x is the right-hand side of a premise. An argument $i \in \mathbb{N}$ of an operator o is *receiving* if a variable x is receiving in a term t that has a subterm $o(v_1, \dots, v_n)$ with x occurring in v_i .
- A rule of the form of
$$\frac{x_i \xrightarrow{\tau} y}{o(x_1, \dots, x_n) \xrightarrow{\tau} o(x_1, \dots, x_n)[y/x_i]}$$
 for $1 \leq i \leq n$ is called a *patience rule* for the i th argument of o , where $t[y/x]$ stands for term t with all occurrences of x replaced by y .

We can now give the complete definition of the simply WB cool rule format.

► **Definition A.1.** A GSOS language \mathcal{L} is simply WB cool if it is positive and the following conditions are all true.

1. All operators in \mathcal{L} are straight.
2. The only rules in \mathcal{L} with τ -premises are patience rules.
3. Every active argument of an operator has a patience rule.
4. Every receiving argument of an operator has a patience rule.
5. All operators in \mathcal{L} are smooth.

B Selected proofs

In this section of the appendix we include the proofs of Proposition 3.1 and Theorem 3.16.

B.1 Equivalence of the two representations of the continuity criterion

Proof of Proposition 3.1. We introduce the friendlier notation $\bar{\Sigma}f$ in place of $\lambda \circ \Sigma^*\langle 1, f \rangle$ for any $f : X \rightarrow TX$. (1) \implies (2) is immediate since $\rho = \lambda \circ \theta$, where $\theta : \Sigma \implies \Sigma^*$ is the universal natural transformation sending Σ to its free monad. For (2) \implies (1), we first note that for each “link” f_i , $\bar{\Sigma}f_i$ is (equivalently) defined via “structural recursion with accumulators” (see [35, Theorem 5.1]), i.e. it is the unique morphism making the following diagram commute.

$$\begin{array}{ccc}
 \Sigma\Sigma^*X & \xrightarrow{\mu \circ \theta_{\Sigma^*}} & \Sigma^*X \xleftarrow{\eta} X \\
 \downarrow \Sigma\langle 1, \bar{\Sigma}f_i \rangle & & \bar{\Sigma}f_i \downarrow \exists! \swarrow T\eta \circ f_i \\
 \Sigma(\text{Id} \times T)\Sigma^*X & \xrightarrow{\rho_{\Sigma^*}} T\Sigma^*\Sigma^*X \xrightarrow{T\mu} T\Sigma^*X &
 \end{array}$$

This makes $\bar{\Sigma}f_i$ the (necessarily unique) *homomorphic extension* of $T\mu \circ \rho_{\Sigma^*}$ along $T\eta \circ f_i$. Continuity of composition in $\mathcal{Kl}(T)$ gives

$$T\eta \circ \bigvee_i f_i = \bigvee_i (T\eta \circ f_i) \quad (13)$$

$T\eta \circ f_i = \bar{\Sigma}f_i \circ \eta$, thus $\bigvee_i (\bar{\Sigma}f_i \circ \eta)$ exists and also

$$\bigvee_i (\bar{\Sigma}f_i \circ \eta) = \left(\bigvee_i \bar{\Sigma}f_i \right) \circ \eta \quad (14)$$

Via similar reasoning, we have

$$T\mu \circ \bigvee_i (\rho_{\Sigma^*} \circ \Sigma\langle 1, \bar{\Sigma}f_i \rangle) = \bigvee_i (T\mu \circ \rho_{\Sigma^*} \circ \Sigma\langle 1, \bar{\Sigma}f_i \rangle) \quad (15)$$

$$\bigvee_i (\bar{\Sigma}f_i \circ \mu \circ \theta_{\Sigma^*}) = \left(\bigvee_i \bar{\Sigma}f_i \right) \circ \mu \circ \theta_{\Sigma^*} \quad (16)$$

Equation (2) allows us to rewrite Equation (15) as

$$T\mu \circ \rho_{\Sigma^*} \circ \Sigma\langle 1, \bigvee_i \bar{\Sigma}f_i \rangle = \bigvee_i (T\mu \circ \rho_{\Sigma^*} \circ \Sigma\langle 1, \bar{\Sigma}f_i \rangle) \quad (17)$$

By taking the supremum over i of the i -dependent arrows in the previous diagram, Equation (13), (14), (16) and (17) allow us to present $\bigvee_i \bar{\Sigma} f_i$ as a morphism that makes the following diagram commute.

$$\begin{array}{ccccc}
 \Sigma \Sigma^* X & \xrightarrow{\mu \circ \theta_{\Sigma^*}} & \Sigma^* X & \xleftarrow{\eta} & X \\
 \downarrow \Sigma \langle 1, \bigvee_i \bar{\Sigma} f_i \rangle & & \downarrow \bigvee_i \bar{\Sigma} f_i & & \swarrow T\eta \circ \bigvee_i f_i \\
 \Sigma(\text{Id} \times T)\Sigma^* X & \xrightarrow{\rho_{\Sigma^*}} & T\Sigma^* \Sigma^* X & \xrightarrow{T\mu} & T\Sigma^* X
 \end{array}$$

But there can only be one such morphism, namely $\bar{\Sigma}(\bigvee_i f_i)$, the unique homomorphic extension of $T\mu \circ \rho_{\Sigma^*}$ along $T\eta \circ \bigvee_i f_i$. In other words, $\bigvee_i \bar{\Sigma} f_i = \bar{\Sigma}(\bigvee_i f_i)$. ◀

B.2 Correspondence with the simply WB cool rule format

Proof of Theorem 3.16. In order to prove that any language \mathcal{L} in the simply WB cool format automatically satisfies the three criteria, it suffices to show that (the GSOS law induced by) any arbitrary rule in \mathcal{L} satisfies them. First of all, we know that rules in the simply WB cool format are of the form $\frac{\{x_i \xrightarrow{c_i} y_i \mid i \in I\}}{o(x_1, \dots, x_n) \xrightarrow{\alpha} t}$ for $I \subseteq \{1, \dots, n\}$ [37, §3], or

simply $\frac{\{x_i \xrightarrow{c_i} y_i \mid i \in I\}}{o(\vec{x}) \xrightarrow{\alpha} t}$, where o is an n -ary operator. We thus consider an arbitrary rule in the above form and proceed by distinguishing by the number of active arguments.

B.2.1 0 active arguments

All cases are trivial.

B.2.2 1 active argument

The rule is of the form $\frac{x_j \xrightarrow{c} y}{o(\vec{x}) \xrightarrow{\alpha} t}$ meaning that the rule is active on a position j . There is only a single premise because of the first requirement (straightness) in Definition A.1.

B.2.2.1 Patience rule

If $c = \tau$ then by the second requirement of Definition A.1 this rule has to be a patience rule

$$\text{of the form } \frac{x_j \xrightarrow{\tau} y}{o(\vec{x}) \xrightarrow{\tau} o(\vec{x})[y/x_j]}.$$

Criterion 1.

$$\begin{aligned}
 \bigvee_i \rho(o(\vec{x}), f_i(\vec{x})) &= \bigvee_i \{\tau, o(\vec{x})[y/x_j] \mid (\tau, y_i) \in f_i(x_j)\} = \\
 \{\tau, o(\vec{x})[y/x_j] \mid (\tau, y_i) \in \bigvee_i (f_i(x_j))\} &= \rho(o(\vec{x}), \bigvee_i (f_i(\vec{x})))
 \end{aligned}$$

Criterion 2. $\rho(o(\vec{x}), (\eta \vee f)(\vec{x}))$ induces a single transition $o(\vec{x}) \xrightarrow{\tau} o(\vec{x})$ for all f , which is included in $\eta_x \circ \theta_X$.

Criterion 3. The only transitions in $\rho(o(\vec{x}, (f \diamond f)(\vec{x})))$ are $o(\vec{x}) \xrightarrow{\tau} o(\vec{x})[z/x_j]$ when $x_j \xrightarrow{\tau} y \xrightarrow{\tau} z$, i.e. $(\tau, y) \in f(x_j), (\tau, z) \in g(y)$ for some y, z . Running $(\lambda_X \circ \Sigma^*(1, f)) \diamond (\rho_X \circ \Sigma(1, f))$ ², which is the second iteration on the right side, we can see that $o(\vec{x}) \xrightarrow{\tau} o(\vec{x})[y/x_j] \xrightarrow{\tau} o(\vec{x})[z/y]$, which satisfies the criterion.

B.2.2.2 Impatient rule

This time we have $c \neq \tau$ which entails, by the third requirement of Definition A.1, the presence of a patience rule for argument j .

Criterion 1. Similar to B.2.2.1.

$$\begin{aligned} \bigvee_i \rho(o(\vec{x}, f_i(\vec{x}))) &= \bigvee_i \{c, o(\vec{x})[y/x_j] \mid (c, y_i) \in f_i(x_j)\} = \\ &\{c, o(\vec{x})[y/x_j] \mid (c, y_i) \in \bigvee_i (f_i(x_j))\} = \rho(o(\vec{x}, \bigvee_i (f_i(\vec{x}))) \end{aligned}$$

Criterion 2. Similarly to B.2.2.1, the presence of the patience rule for argument j means that there is always transition $o(\vec{x}) \xrightarrow{\tau} o(\vec{x})$ for all f on the left side, which is included on the right side by $\eta_x \circ \theta_X$. The transitions induced by f exist on both sides.

Criterion 3. Transitions on the left side occur if and only if there are w, z such that $x_j \xrightarrow{\tau} w \xrightarrow{c} z$ or $x_j \xrightarrow{c} w \xrightarrow{\tau} z$, i.e. $(\tau, w) \in f(x_j), (c, z) \in f(w)$ or $(c, w) \in f(x_j), (\tau, z) \in f(w)$. We also have to distinguish between y in premise $x_j \xrightarrow{c} y$ being receiving in t or not. For each case, we give the transition(s) on the left side (of Criterion 3) and the respective iteration step where the left-side transitions appear on the right side (of Criterion 3).

1. y is not receiving, $x_j \xrightarrow{\tau} w \xrightarrow{c} z$.
 - $\rho_X \circ \Sigma(1, f \diamond f)$: $o(\vec{x}) \xrightarrow{\alpha} t$
 - $(\lambda_X \circ \Sigma^*(1, f)) \diamond (\rho_X \circ \Sigma(1, f))$: $o(\vec{x}) \xrightarrow{\tau} o(\vec{x})[w/x_j] \xrightarrow{\alpha} t$
2. y is not receiving, $x_j \xrightarrow{c} w \xrightarrow{\tau} z$.
 - $\rho_X \circ \Sigma(1, f \diamond f)$: $o(\vec{x}) \xrightarrow{\alpha} t$
 - $\rho_X \circ \Sigma(1, f)$: $o(\vec{x}) \xrightarrow{\alpha} t$
3. y is receiving, $x_j \xrightarrow{\tau} w \xrightarrow{c} z$.
 - $\rho_X \circ \Sigma(1, f \diamond f)$: $o(\vec{x}) \xrightarrow{\alpha} t(z)$
 - $(\lambda_X \circ \Sigma^*(1, f)) \diamond (\rho_X \circ \Sigma(1, f))$: $o(\vec{x}) \xrightarrow{\tau} o(\vec{x})[w/x_j] \xrightarrow{\alpha} t(z)$
4. y is receiving, $x_j \xrightarrow{c} w \xrightarrow{\tau} z$

This is the trickiest case. Let us look back at the rule in question (with y receiving), which is $\frac{x_j \xrightarrow{c} y}{o(\vec{x}) \xrightarrow{\alpha} t(y)}$. The key observation is that requirement 4 in Definition A.1,

requiring that every receiving argument of an operator has a patience rule, implies that no matter how complex the receiving expression $t(y)$ is, there will be patience rules in place to ensure a derivation amounting to $\frac{x \xrightarrow{\tau} y}{s(x) \xrightarrow{\tau} s(y)}$ for each sub-expression $s(y)$

in $t(y)$ that “receives” y . The number of iterations on the right-hand required in order to trigger all necessary patience rules depends on the number of sub-expressions s .

For example, let $t(y) \triangleq d(e(l(y, 0)), e(l(0, y)))$, where d, l are binary operations, e is a unary operation and 0 is some term. Due to requirement 4, d, l, e will all have patience

² Recall that $(\lambda_X \circ \Sigma^*(1, f)) \circ \theta = \rho_X \circ \Sigma(1, f)$.

rules in all positions and we need exactly two iterations to trigger the two patience rules in each of the positions of d . More generally,

- $\rho_X \circ \Sigma\langle 1, f \diamond f \rangle: o(\vec{x}) \xrightarrow{\alpha} t(z)$
- $(\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \diamond (\rho_X \circ \Sigma\langle 1, f \rangle): o(\vec{x}) \xrightarrow{\alpha} t(w) \xrightarrow{\tau^*} t(z)$.

B.2.3 2 or more active arguments

We first note that the only rules with τ -premises are patience rules, which are already covered in B.2.2.1 and so we move on to $c_1, c_2 \neq \tau$ with the rule being of the form of

$$\frac{x_i \xrightarrow{c_1} y_1 \quad x_j \xrightarrow{c_2} y_2}{o(\vec{x}) \xrightarrow{\alpha} t}$$

The third requirement of Definition A.1 means that there are patience rules for arguments i and j in o .

Criterion 1. Similar to the case for Criterion 1 in B.2.2.1.

Criterion 2. Similar to the case for Criterion 2 in B.2.2.1.

Criterion 3. There are four separate cases where a transition occurs in $\rho(o(\vec{x}, (f \diamond f)(\vec{x})))$, as a c_1 -transition and a c_2 -transition may occur in either step for both subterms.

1. $x_i \xrightarrow{\tau} w_1 \xrightarrow{c_1} z_1$ and $x_j \xrightarrow{\tau} w_2 \xrightarrow{c_2} z_2$
2. $x_i \xrightarrow{\tau} w_1 \xrightarrow{c_1} z_1$ and $x_j \xrightarrow{c_2} w_2 \xrightarrow{\tau} z_2$
3. $x_i \xrightarrow{c_1} w_1 \xrightarrow{\tau} z_1$ and $x_j \xrightarrow{c_2} w_2 \xrightarrow{\tau} z_2$
4. $x_i \xrightarrow{c_1} w_1 \xrightarrow{\tau} z_1$ and $x_j \xrightarrow{\tau} w_2 \xrightarrow{c_2} z_2$.

In addition, y_1 and y_2 can each be either receiving or not receiving in t and, as this affects transitions the same way as y being receiving in B.2.2.2.

1. $x_i \xrightarrow{\tau} w_1 \xrightarrow{c_1} z_1$ and $x_j \xrightarrow{\tau} w_2 \xrightarrow{c_2} z_2$

Here, whether y_1 and y_2 are receiving or not does not make a difference and we write $t(y_1, y_2)$ to denote a term which potentially has instances of y_1 and y_2 .

- $\rho_X \circ \Sigma\langle 1, f \diamond f \rangle: o(\vec{x}) \xrightarrow{\alpha} t(z_1, z_2)$
- $(\lambda_X \circ \Sigma^*\langle 1, f \rangle) \diamond (\lambda_X \circ \Sigma^*\langle 1, f \rangle) \diamond (\rho_X \circ \Sigma\langle 1, f \rangle):$
 $o(\vec{x}) \xrightarrow{\tau} o(\vec{x})[w_1/x_i] \xrightarrow{\tau} o(\vec{x})[w_1/x_i][w_2/x_j] \xrightarrow{\alpha} t(z_1, z_2)$

We can see that we need to iterate three times on the right-hand side: one to trigger the patience rule on position i , one to trigger the patience rule on position j on the new term and one more to trigger the main rule.

2. $x_i \xrightarrow{\tau} w_1 \xrightarrow{c_1} z_1$ and $x_j \xrightarrow{c_2} w_2 \xrightarrow{\tau} z_2$

In this case y_2 being receiving makes a difference, while y_1 does not. Let us first deal with the non-receiving case for y_2 .

- $\rho_X \circ \Sigma\langle 1, f \diamond f \rangle: o(\vec{x}) \xrightarrow{\alpha} t(z_1)$
- $(\lambda_X \circ \Sigma^*\langle 1, f \rangle) \diamond (\rho_X \circ \Sigma\langle 1, f \rangle): o(\vec{x}) \xrightarrow{\tau} o(\vec{x})[w_1/x_i] \xrightarrow{\alpha} t(z_1)$

The first iteration will trigger the patience rule for i , while the second produces the α -transition. Variable y_2 is not receiving and so nothing else is needed. On the other hand, if y_2 is receiving, we see that

- $\rho_X \circ \Sigma\langle 1, f \diamond f \rangle: o(\vec{x}) \xrightarrow{\alpha} t(z_1, z_2)$
- $(\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \diamond (\lambda_X \circ \Sigma^*\langle 1, f \rangle) \diamond (\rho_X \circ \Sigma\langle 1, f \rangle):$
 $o(\vec{x}) \xrightarrow{\tau} o(\vec{x})[w_1/x_i] \xrightarrow{\alpha} t(z_1, w_2) \xrightarrow{\tau^*} t(z_1, z_2)$

Similarly to the fourth case of Criterion 3 in B.2.2.2, requirement 4 in Definition A.1 guarantees that there will be a sequence of patience rules that gives $t(z_1, w_2) \xrightarrow{\tau^*} t(z_1, z_2)$.

$$3. x_i \xrightarrow{c_1} w_1 \xrightarrow{\tau} z_1 \text{ and } x_j \xrightarrow{c_2} w_2 \xrightarrow{\tau} z_2$$

If y_1 and y_2 are not receiving, this is very similar to the first case. If y_1 and/or y_2 are receiving, then requirement 4 in Definition A.1 comes into play in the same manner as before. For instance, if both y_1 and y_2 are receiving:

$$\begin{aligned} &= \rho_X \circ \Sigma\langle 1, f \diamond f \rangle: o(\vec{x}) \xrightarrow{\alpha} t(z_1, z_2) \\ &= (\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \diamond (\lambda_X \circ \Sigma^*\langle 1, f \rangle)^* \diamond (\rho_X \circ \Sigma\langle 1, f \rangle): \\ &\quad o(\vec{x}) \xrightarrow{\alpha} t(w_1, w_2) \xrightarrow{\tau^*} t(z_1, w_2) \xrightarrow{\tau^*} t(z_1, z_2) \end{aligned}$$

$$4. x_i \xrightarrow{c_1} w_1 \xrightarrow{\tau} z_1 \text{ and } x_j \xrightarrow{\tau} w_2 \xrightarrow{c_2} z_2$$

Very similar to the second case.

In the presence of three or more active arguments we can apply the same principles, the only difference being that the maximum number of necessary iterations on the right side will be higher, as more patience rules will have to be triggered. ◀

Quantum Multiple-Valued Decision Diagrams in Graphical Calculi

Renaud Vilmart   

Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria,
Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Abstract

Graphical calculi such as the ZH-calculus are powerful tools in the study and analysis of quantum processes, with links to other models of quantum computation such as quantum circuits, measurement-based computing, etc.

A somewhat compact but systematic way to describe a quantum process is through the use of quantum multiple-valued decision diagrams (QMDDs), which have already been used for the synthesis of quantum circuits as well as for verification.

We show in this paper how to turn a QMDD into an equivalent ZH-diagram, and vice-versa, and show how reducing a QMDD translates in the ZH-Calculus, hence allowing tools from one formalism to be used into the other.

2012 ACM Subject Classification Theory of computation → Quantum information theory; Theory of computation → Equational logic and rewriting

Keywords and phrases Quantum Computing, ZH-Calculus, Decision Diagrams

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.89

Related Version *Full Version:* <https://hal.archives-ouvertes.fr/hal-03277262>

1 Introduction

Graphical calculi for quantum computation such as the ZX-Calculus [9], the ZW-Calculus [10] and the ZH-Calculus [2] are powerful yet intuitive tools for the design and analysis of quantum processes. They have already been successfully applied to the study of measurement-based quantum computing [15], error correction through the operations of lattice surgery on surface codes [12, 13], as well as for the optimisation of quantum circuits [4, 11, 22]. Their strong links with “sums-over-paths” [1, 23, 28], as well as their respective complete equational theories [4, 16, 21, 27], make them good candidates for automated verification [7, 14, 17].

An important question, whose answer benefits a lot of these different aspects, is the one of synthesis. Given a description of a quantum process, how do we turn it into a ZX-diagram? This all depends on the provided description. It was already shown how to efficiently get a diagram from quantum circuits [4], from a measurement-based process [15], from a sequence of lattice surgery operations [13], from “sums-over-paths” [23], or even from the whole matrix representation of the process [20]. Although this last translation is efficient in the size of the matrix, the size of the matrix itself grows exponentially in the number of qubits, so few processes will actually be given in terms of their whole matrix.

The matrix representation however has an advantage: it is (essentially) unique. Two quantum operators are operationally the same if and only if their matrix representations are colinear. This is to be contrasted with all the different previous examples, where for instance two different quantum circuits may implement the same operator.

The form of the ZX-diagram obtained from a quantum state by [20] is that of a binary tree: a branching in the tree corresponds to a cut in half of the represented vector, while the leaves of the tree exactly correspond to the entries in the vector. It is however possible to exploit redundancies in the entries of the vector, by merging similar subtrees. Doing so alters



© Renaud Vilmart;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 89; pp. 89:1–89:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the notion of normal form by compacting it, whilst retaining its uniqueness property. This can be done at the level of the ZX-diagram using its equational theory, and in particular some equality that is reminiscent to that of a bialgebra rule. Doing this from a proper tree on the other hand gives rise to a quantum version of a decision diagram, which has already been introduced in [24]. The so-called quantum multiple-valued decision diagrams (QMDDs) [25] have since then been used to synthesise quantum circuits [26] or to perform verification of quantum programs [5, 6].

We hence aim in this paper at showing the links between the aforementioned graphical calculi and QMDDs. We in particular show how to translate from one formalism to the other, and how the reduction of a QMDD translates in the graphical languages. As a consequence, tools developed in one formalism may be transported and used in the other. Additionally, this result together with the aforementioned results in the graphical languages, relates the QMDDs to measurement-based computation, lattice surgery operations, “sums-over-paths”, etc.


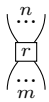




In Section 2, we present the ZH-calculus, the graphical language we will use in this paper for convenience. We then present in Section 3 the quantum multiple-valued decision diagrams. In Section 4, we show how to turn a QMDD into a ZH-diagram that represents the same quantum operator. In Section 5, we show an algorithm to turn a ZH-diagram into QMDD form, which can be used to get the QMDD description of any ZH-diagram.

2 The ZH-Calculus

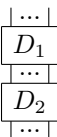
We aim in this paper at showing links between quantum multiple-valued decision diagrams and graphical languages for quantum computing: ZX, ZW and ZH. Since any of the three languages can be translated in the other two [2, 16, 19], we may simply choose one. It so happens that the closest to QMDDs we have is the ZH-Calculus. We hence present here this language.

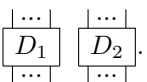
2.1 ZH-Diagrams

A ZH-diagram $D : k \rightarrow \ell$ with k inputs and ℓ outputs is generated by:

- $Z_m^n : n \rightarrow m ::$  called Z-spiders
- $H_m^n(r) : n \rightarrow m ::$  called H-spiders
- $id : 1 \rightarrow 1 ::$ 
- $\sigma : 2 \rightarrow 2 ::$ 
- $\eta : 0 \rightarrow 2 ::$ 
- $\epsilon : 2 \rightarrow 0 ::$ 

where $n, m \in \mathbb{N}$ and $r \in \mathbb{C}$. In the following, we may write H-spiders with no parameter, in which case, the implied parameter is -1 by convention.

Diagrams can then be composed either sequentially:  (if the number of output of the

top diagram matches the number of inputs of the bottom one), or in parallel: .

It is customary to define the additional two “X-spiders”:

$$\begin{array}{c} \dots \\ \vdots \\ \bullet \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} := \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \quad \text{and} \quad \begin{array}{c} \dots \\ \vdots \\ \neg \bullet \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} := \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array}$$

ZH-diagrams can be understood as quantum operators thanks to the *standard interpretation* $\llbracket \cdot \rrbracket$ which maps any ZH-diagram $D : n \rightarrow m$ to a complex matrix $\llbracket D \rrbracket \in \mathbb{C}^{2^m} \times \mathbb{C}^{2^n}$, and which is inductively defined as:

$$\begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} = \llbracket D_2 \rrbracket \circ \llbracket D_1 \rrbracket \quad \llbracket \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \rrbracket = \llbracket D_1 \rrbracket \otimes \llbracket D_2 \rrbracket$$

$$\llbracket \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \rrbracket = 2^m \begin{array}{c} \overbrace{\begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & 0 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & 0 & 0 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix}}^{2^n} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{array} \quad \llbracket \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \rrbracket = 2^m \begin{array}{c} \overbrace{\begin{pmatrix} 1 & \dots & \dots & 1 \\ \vdots & \ddots & & \vdots \\ \vdots & & 1 & 1 \\ 1 & \dots & 1 & r \end{pmatrix}}^{2^n} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{array}$$

$$\llbracket | \rrbracket = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \llbracket \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \rrbracket = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \llbracket \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \rrbracket = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The standard interpretation of the X-spiders can then be obtained by composition. We underline that $\llbracket \begin{array}{c} \dots \\ \vdots \\ \bullet \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \rrbracket = |0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and that $\llbracket \begin{array}{c} \dots \\ \vdots \\ \neg \bullet \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \rrbracket = |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

We have used here the Dirac notation, where a quantum state i.e. a vector is denoted $|\psi\rangle$. We recall that $\langle \psi |$ is defined as $|\psi\rangle^\dagger$ where $(\cdot)^\dagger$ yields the transconjugate of a matrix.

The language is universal, i.e. any quantum operator can be represented as a ZH-diagram [2]:

$$\forall f \in \mathbb{C}^{2^m} \times \mathbb{C}^{2^n}, \exists D \in \mathbf{ZH}(n, m), \llbracket D \rrbracket = f$$

An important result that we will use in the following is the fact that there is an isomorphism between $\mathbf{ZH}(n, m)$ and $\mathbf{ZH}(0, m + n)$, i.e. any ZH-diagram $D : n \rightarrow m$ can be turned into a *state* $D' : 0 \rightarrow m + n$ in a reversible manner. This is called the *map/state duality* [9, 8, 18].

Graphically, this isomorphism is obtained by $\psi_{n,m}(D) = \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array}$ and $\psi_{n,m}^{-1}(D') = \begin{array}{c} \dots \\ \vdots \\ \square \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array}$,

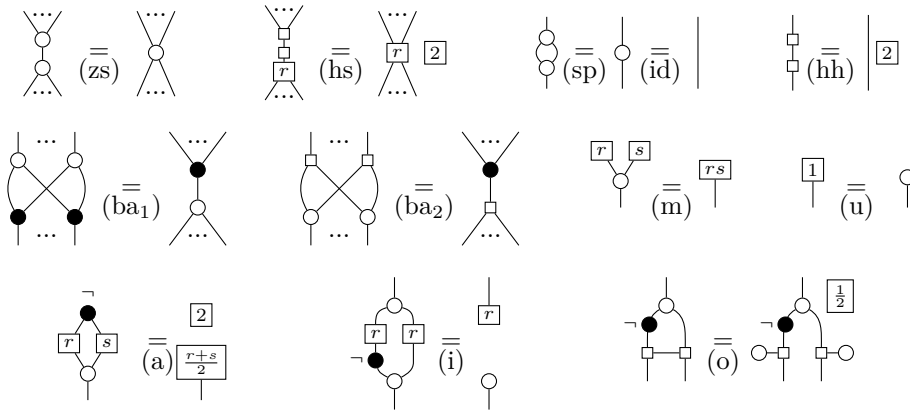
and the fact that this is indeed an isomorphism comes from the fact that:

$$\begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} = \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \quad \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} = \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array} \begin{array}{c} \dots \\ \vdots \\ \vdots \\ \vdots \\ \dots \end{array}$$

Notice that this definition of the map/state duality differs from more usual ones by a rearranging of the wires. This is useful in the following to better relate state-QMDDs to proper QMDDs.

2.2 Equational Theory

The previous equalities constitute the first of a series of axioms that makes up an equational theory for the language. The axioms are summed up in Figure 1.



■ **Figure 1** Rules of the ZH-Calculus.

The previous equality is actually part of an implicit set of axioms of the language, aggregated under the paradigm “only connectivity matters”, which states that all deformations of diagrams are allowed.

When we can turn a diagram D_1 into another diagram D_2 by a succession of the transformations in Figure 1, we usually write $ZH \vdash D_1 = D_2$, however, to keep things simple, we will abbreviate it as $D_1 = D_2$ in this paper. This set of rules was proven to be sound and complete [2], that is:

$$\forall D_1, D_2 \in \mathbf{ZH}(n, m), \llbracket D_1 \rrbracket = \llbracket D_2 \rrbracket \iff ZH \vdash D_1 = D_2$$

Two useful lemmas that will be important in the following can be derived from the previous axiomatisation, as proved in [3]:

► **Lemma 1.**

$$\begin{array}{c} \dots \\ \circ \\ \hline 0 \\ \hline \circ \\ \dots \end{array} = \frac{1}{2} \begin{array}{c} \dots \\ \circ \\ \hline \circ \\ \hline \circ \\ \dots \end{array}$$

► **Lemma 2.**

$$\begin{array}{c} \hline 0 \\ \hline \circ \end{array} = \bullet$$

In the following, since $\llbracket r \rrbracket = r$ and since it represents a global scalar, we may “forget the box”, and write $\begin{array}{c} \dots \\ \hline r \\ \hline D \\ \hline \dots \end{array}$ simply as $r \begin{array}{c} \dots \\ \hline D \\ \hline \dots \end{array}$. The equation $\begin{array}{c} \hline r \\ \hline s \end{array} = \begin{array}{c} \hline rs \end{array}$ is also derivable [3], which means that following the convention, different global scalars will get multiplied. For instance, we have $(r_1 \cdot D_1) \otimes (r_2 \cdot D_2) = (r_1 r_2) \cdot (D_1 \otimes D_2)$.



2.3 New Constructions

To make the link between decision diagrams and ZH-diagrams, we feel it is simpler to

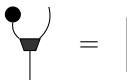
introduce two new constructions: $\begin{array}{c} \hline \square \\ \hline \square \end{array} := \frac{1}{2} \begin{array}{c} \hline \square \\ \hline \square \\ \hline \bullet \end{array}$ and $\begin{array}{c} \hline \cup \\ \hline \end{array} := \begin{array}{c} \hline 0 \\ \hline \circ \\ \hline \bullet \end{array} = \frac{1}{2} \begin{array}{c} \hline \square \\ \hline \square \\ \hline \bullet \end{array}$

of which we may compute the standard interpretation:

$$\llbracket \begin{array}{c} \hline \cup \\ \hline \end{array} \rrbracket = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } \llbracket \begin{array}{c} \hline \square \\ \hline \square \end{array} \rrbracket = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

 corresponds to the diagram  of the ZW-Calculus [10], it is very close to the so-called W-state. It can also be found in [3] in the context of the ZH-calculus. The interaction of this building block with classical states is given by:

► **Lemma 3.**



► **Lemma 4.**

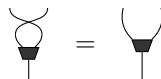


The pair $(\text{triangle with dot}, \text{dot})$ can be seen as a commutative monoid, which means that, on top of Lemma 3, the following is true:

► **Lemma 5.**



► **Lemma 6.**



Proof. Proof for associativity can be found in [3], and commutativity is obvious from the definition of the operator. ◀

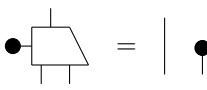
This allows us to use a generalised version of this diagram, with arbitrary number of inputs, defined inductively as follows: $\text{triangle with dot} := \text{dot}$ and $\text{triangle with dot} := \text{triangle with dot}$. In practice, in the following, we will not use the case 0 inputs. We will however use the case 1 extensively, which we will assume simplified: $\text{triangle with dot} = \text{vertical line}$

The second diagram is the ZH version of the gadget used in the normal forms of [20], and it can be understood as follows:

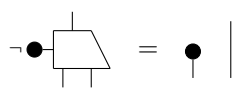
$$\left[\text{trapezoid with dot} \right] = \langle 0 | \otimes \left[\text{vertical line with dot} \right] + \langle 1 | \otimes \left[\text{vertical line with dot} \right]$$

so it is a diagram that builds $\text{vertical line with dot}$ and whose leftmost wire controls whether or not the outputs are swapped. This can be checked diagrammatically:

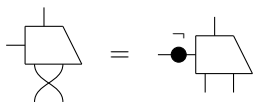
► **Lemma 7.**



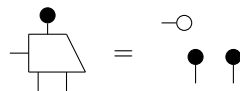
► **Lemma 9.**



► **Lemma 8.**



► **Lemma 10.**



3 Quantum Multiple-valued Decision Diagrams

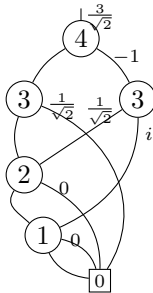
Quantum multiple-valued decision diagrams (QMDD) were introduced to store quantum unitaries in a way that is efficient in certain cases, similarly to binary decision diagrams for representing decision problems. In the following, we use the map/state duality to turn every map into a state. As a consequence, QMDDs are adapted to “state-QMDDs” as follows:

- **Definition 11.** A state-QMDD (SQMDD) is a tuple $(s, V, u_0, t, H, h, E, \omega)$ where:
- $s \in \mathbb{C}$ is called the overall scalar
 - $V \neq \emptyset$ is a set of vertices
 - $u_0, t \in V$ are two distinguished vertices, called respectively root and terminal node
 - $u_0 = t \iff V = \{t\}$, i.e. u_0 and t coincide only if V only contains one vertex
 - $H \in \mathbb{N}$ is the height of the SQMDD
 - $h : V \mapsto \{0, \dots, H\}$ maps each vertex to their height in the SQMDD
 - $h(u) = 0 \iff u = t$
 - $E : V \setminus \{t\} \rightarrow V^2$ maps any non-terminal vertex to a pair of vertices. These are the edges of the SQMDD
 - If $E(u) = (v_0, v_1)$ then $h(v_i) < h(u)$ for $i \in \{0, 1\}$
 - $\forall v \in V \setminus \{u_0\}, \exists u \in V, v \in E(u)$, i.e. all vertices have at least one parent
 - $\omega : V \setminus \{t\} \rightarrow \mathbb{C}^2$ maps edges to complex weights




Notice that the requirement on the heights of two endpoints of an edge also enforces the fact that an SQMDD is acyclical.

When drawing a SQMDD, it is relevant to set all the same-height nodes at the same height in the representation. It is also customary to omit writing weights of 1, as well as the vertices’ names (we write instead their height). We highlight the root by an incoming wire, and distinguish the terminal node by drawing it as a square, instead of a circle for the other nodes (following [24]’s convention). Finally, we display the overall scalar as a weight on the incoming edge of the root, and if $h(u_0) = H$, we omit H , which we otherwise specify at the top of the SQMDD.

► **Example 12.** The diagram:



is a graphical representation of an SQMDD.

► **Remark 13.** SQMDDs can be seen as a more fine-grained version of proper QMDDs. Indeed, using the map/state duality $\psi_{n,m}$ defined above amounts to decomposing a QMDD vertex  into . This for instance allows us to apply a swap  only on one side of a circuit/diagram, while the associated QMDD notion of variable reordering requires swapping the whole qubit i.e. apply a swap on both sides of the circuit. A similar presentation of QMDDs for states can be found in [29].

We can define two natural constructions from any SQMDD of height ≥ 1 :

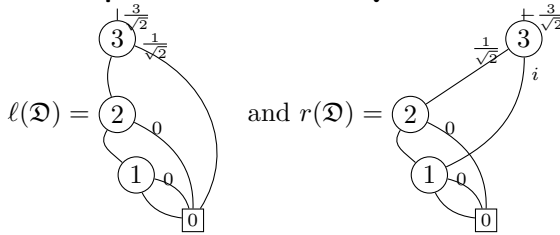
► **Definition 14.** Let $\mathfrak{D} = (s, V, u_0, t, H, h, E, \omega)$ be a SQMDD with $H \geq 1$. We denote $\ell(\mathfrak{D})$ the diagram obtained from \mathfrak{D} by:

- If $h(u_0) = H$:
 - replacing the overall scalar s by $s \cdot \pi_0 \omega(u_0)$
 - removing all nodes (and subsequent edges) that cannot be reached by $\pi_0 E(u_0)$
 - replacing the root u_0 by $\pi_0 E(u_0)$
- decreasing the height H by 1

Where π_0 and π_1 are the usual left and right projectors of a pair.

We can similarly build $r(\mathfrak{D})$ by replacing π_0 by π_1 in the definition.

► **Example 15.** With \mathfrak{D} the SQMDD defined in the previous example:



We can now use these constructions to understand SQMDDs as representations of quantum states.

► **Definition 16.** For any SQMDD \mathfrak{D} , $[[\mathfrak{D}]]$ is the (unnormalised) quantum state inductively defined as:

- $[[\begin{smallmatrix} s \\ 0 \end{smallmatrix}]] = s = s | \rangle$
- $[[\mathfrak{D}]] = |0\rangle \otimes [[\ell(\mathfrak{D})]] + |1\rangle \otimes [[r(\mathfrak{D})]]$

(here $| \rangle$ is used to represent the vector (1) i.e. the canonical 0-qubit state).

► **Example 17.** With \mathfrak{D} the diagram of example 12:

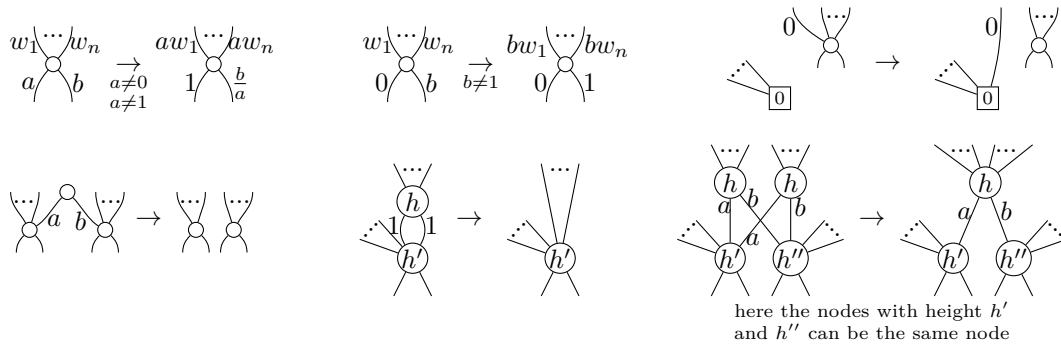
$$[[\mathfrak{D}]] = \frac{3}{\sqrt{2}} \left(1 \ 0 \ 0 \ 0 \ \frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}} \ -\frac{1}{\sqrt{2}} \ 0 \ 0 \ 0 \ -i \ 0 \ -i \ 0 \right)^T$$

The definition of SQMDDs given here does not only differ from that of [24] by the fact that we only consider states, but also by the fact that our definition is laxer. As a consequence, different SQMDDs can have the same interpretation. To address this problem, we can give a set of rewrite rules that will reduce the “size” of the SQMDD while preserving its interpretation. We give in Figure 2 this set of rewrite rules, expressed graphically. Notice that the rules use the graphical notation to encompass transformations on the root and the overall scalar.

It is fairly easy to see that this rewriting terminates. Let us denote $\text{deg}(t)$ the arity of t , i.e. the number of occurrences of t in $E(V \setminus \{t\})$. Notice that $\text{deg}(t) < 2|V|$. For $u \in V \setminus \{t\}$, define:

$$\delta(u) := \begin{cases} 0 & \text{if } \pi_0(\omega(u)) = 1 \vee (\pi_0(\omega(u)) = 0 \wedge \pi_1(\omega(u)) \in \{0, 1\}) \\ 1 & \text{otherwise} \end{cases}$$

Consider now for any SQMDD the quantity $\left(|V|, 2|V| - \text{deg}(t), \sum_{u|h(u)=1} \delta(u), \dots, \sum_{u|h(u)=H} \delta(u) \right)$, and the lexicographical order over these. We can see that this quantity is reduced by any rewrite.



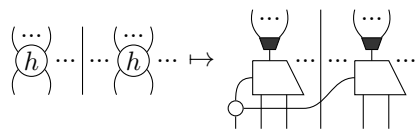
■ **Figure 2** Simplification rules for QMDDs.

When none of these rewrite rules can be applied on an SQMDD, it is called irreducible. Notice that in an irreducible SQMDD, using the notions of [24], no non-terminal vertex is redundant, and all non-terminal vertices are normalised and unique. Hence, an irreducible SQMDD is what [24] properly calls a QMDD, from which we get:

► **Theorem 18** ([24]). *For any quantum state $|\psi\rangle \in \mathbb{C}^{2^n}$, there exists a unique irreducible SQMDD \mathfrak{D} (of height n) such that $\llbracket \mathfrak{D} \rrbracket = |\psi\rangle$.*

4 From SQMDDs to ZH

From any SQMDD, it is possible to build a ZH-diagram whose interpretation will be the same, in a fairly straightforward manner, using the two syntactic sugars defined above. We denote $[\cdot]^{ZH}$ this map. We define it on every “layer” of an SQMDD, that is, on all the nodes of the same height $\leq H$. Such a layer is mapped to a ZH-diagram as follows:



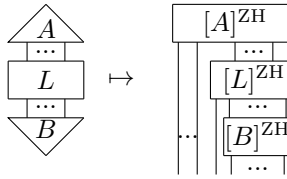
This construction adds a wire to the left. It is the “effective” wire of the layer, the one that will constitute the output qubit in the quantum state. If there is no node of height h , the construction still add a wire on the left, disconnected from everything: \circlearrowleft . Omitted in the

previous definition is the mapping of the weights on wires, which are simply: $r \mapsto \text{gate}(r)$,

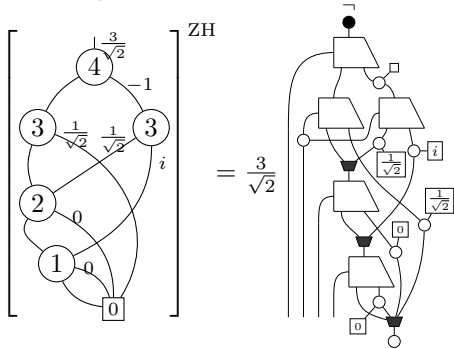
as well that of the terminal node, for which we have: $\text{node}(0) \mapsto \text{terminal node}$. Finally, the particular

state $\text{node}(1)$ is plugged on top of the root: $\text{node}(1) \mapsto \text{gate}(1)$ (if the root has height $< H$ we technically have empty layers on top of the root, hence the $\text{node}(1)$'s).

The full SQMDD is then mapped as follows, where L is the layer of interest, and A and B will themselves be inductively decomposed layer by layer:



► **Example 19.**



This interpretation $[\cdot]^{ZH}$ was not chosen at random: it builds a quantum state with the intended semantics.

► **Proposition 20.** For any SQMDD \mathfrak{D} , $[[\mathfrak{D}]^{ZH}] = [\mathfrak{D}]$.

Proof. Since \bullet and $\neg\bullet$ represent $|0\rangle$ and $|1\rangle$ respectively, we have, for any ZH-diagram $D : 0 \rightarrow n$:

$$\left[\begin{array}{c} D \\ \vdots \end{array} \right] = |0\rangle \otimes \left[\begin{array}{c} D \\ \bullet \end{array} \right] + |1\rangle \otimes \left[\begin{array}{c} D \\ \neg\bullet \end{array} \right]$$

Now, let \mathfrak{D} be an SQMDD, and $D := [\mathfrak{D}]^{ZH}$. We proceed by induction on H the height of \mathfrak{D} , where the base case is obvious. We then focus on the root u_0 . We need to distinguish two cases:

• $H > h(u_0)$: In this case, $\ell(\mathfrak{D}) = r(\mathfrak{D})$ which entails $[\mathfrak{D}] = (|0\rangle + |1\rangle) \otimes [\ell(\mathfrak{D})]$. It can be easily seen that $[\mathfrak{D}]^{ZH} = \bigcirc [\ell(\mathfrak{D})]^{ZH}$, hence $[[\mathfrak{D}]^{ZH}] = \left[\bigcirc \right] \otimes [[\ell(\mathfrak{D})]^{ZH}] = (|0\rangle + |1\rangle) \otimes [[\ell(\mathfrak{D})]^{ZH}]$

• $H = h(u_0)$: Then, $D = \left[\begin{array}{c} \neg\bullet \\ \diagdown \\ D' \\ \vdots \end{array} \right]$. We hence have:

$$\begin{aligned} [[\mathfrak{D}]^{ZH}] &= \left[\begin{array}{c} \neg\bullet \\ \diagdown \\ D' \\ \vdots \end{array} \right] = |0\rangle \otimes \left[\begin{array}{c} \bullet \\ \diagdown \\ D' \\ \vdots \end{array} \right] + |1\rangle \otimes \left[\begin{array}{c} \neg\bullet \\ \diagdown \\ D' \\ \vdots \end{array} \right] \\ &= |0\rangle \otimes \left[\begin{array}{c} \bullet \\ \bullet \\ D' \\ \vdots \end{array} \right] + |1\rangle \otimes \left[\begin{array}{c} \bullet \\ \bullet \\ D' \\ \vdots \end{array} \right] \end{aligned}$$

where the last equality is obtained thanks to Lemmas 7 and 9. We now need to show that

$\llbracket \begin{array}{c} \bullet \bullet \\ \boxed{D'} \\ \vdots \end{array} \rrbracket = \llbracket [\ell(\mathcal{D})]^{ZH} \rrbracket$ and similarly with the right hand side. We can actually show that

we can reduce $\begin{array}{c} \bullet \bullet \\ \boxed{D'} \\ \vdots \end{array}$ to $[\ell(\mathcal{D})]^{ZH}$ using the following rewrites (provable in ZH):

$$\begin{array}{c} \bullet \\ \neg \\ \circ \\ \boxed{r} \\ \vdots \end{array} \rightarrow r \begin{array}{c} \neg \\ \bullet \\ \vdots \end{array} \quad (1) \quad \begin{array}{c} \bullet \\ \diagup \diagdown \\ \square \\ \vdots \end{array} \rightarrow \begin{array}{c} \circ \\ \bullet \bullet \\ \vdots \end{array} \quad (2) \quad \begin{array}{c} \bullet \\ \circ \\ \boxed{r} \\ \vdots \end{array} \rightarrow \begin{array}{c} \bullet \\ \vdots \end{array} \quad (3)$$

$$\begin{array}{c} \bullet \dots \\ \cup \\ \vdots \end{array} \rightarrow \begin{array}{c} \dots \\ \cup \\ \vdots \end{array} \quad (4) \quad \begin{array}{c} \dots \\ \diagup \diagdown \\ \circ \\ \vdots \end{array} \rightarrow \begin{array}{c} \dots \\ \diagdown \diagup \\ \circ \\ \vdots \end{array} \quad (5)$$

Rewrite 1 ensures that if the left wire was weighted, the weight itself gets factored in the overall scalar.

Rewrites 2 and 3 destroy all the nodes that are not descendent of the left child of the root. Rewrite 4 dictates that if \bullet arrives at a node with several parents, the behaviour depends

on what happens to the others parents (if all parents are destroyed, \cup will get \bullet to all its inputs, which will result in \bullet , hence pursuing the destruction of subsequent nodes). It also shows what happens to \bullet when it arrives at the terminal node.

Finally, Rewrite 5 simply normalises the connected Z-spiders one can obtain from Rewrite 2.

This rewrite strategy goes on as long as some \bullet exist in the diagram, until they all disappear from Rewrite 4, in which situation we get the diagram one would have obtained from $\ell(\mathcal{D})$. Similarly, we see that $\begin{array}{c} \bullet \bullet \\ \boxed{D'} \\ \vdots \end{array}$ reduces to $[r(\mathcal{D})]^{ZH}$. Hence, by soundness of the rewrite strategy, we do have:

$$\llbracket [\mathcal{D}]^{ZH} \rrbracket = |0\rangle \otimes \llbracket [\ell(\mathcal{D})]^{ZH} \rrbracket + |1\rangle \otimes \llbracket [r(\mathcal{D})]^{ZH} \rrbracket$$

which by induction hypothesis means $\llbracket [\mathcal{D}]^{ZH} \rrbracket = \llbracket \mathcal{D} \rrbracket$. ◀

This proof introduces a small rewrite strategy, that will be used in the following, in particular to simplify a ZH-diagram in SQMDD form.

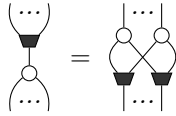
5 Setting a ZH-Diagram in SQMDD Form

If any SQMDD can be turned into a ZH-diagram, the reciprocal requires some work. In the following, we describe an algorithm that turns any ZH-diagram into SQMDD form, i.e. into a ZH-diagram that is in the image of $[\cdot]^{ZH}$, using its equational theory.

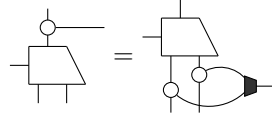
5.1 SQMDD Reduction

We start to show that all the simplification rules for SQMDDs can be derived directly into the ZH-Calculus. For this, we need the following lemmas:

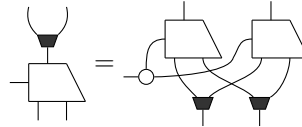
► Lemma 21.



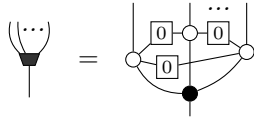
► Lemma 24.



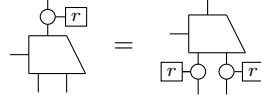
► Lemma 27.



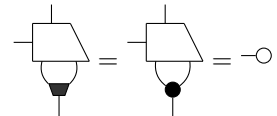
► Lemma 22.



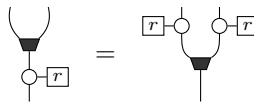
► Lemma 25.



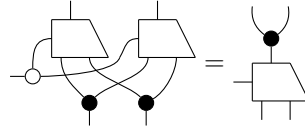
► Lemma 28.



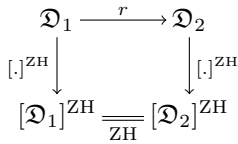
► Lemma 23.



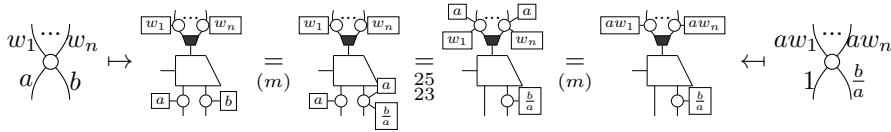
► Lemma 26.



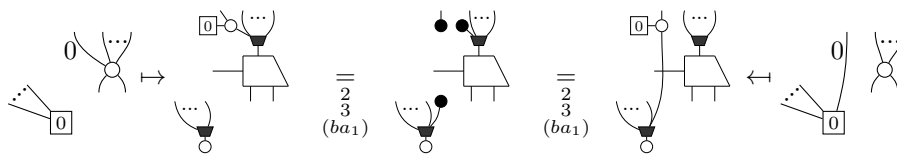
► Proposition 29. For any simplification rule $\mathfrak{D}_1 \xrightarrow{r} \mathfrak{D}_2$, the following diagram commutes:



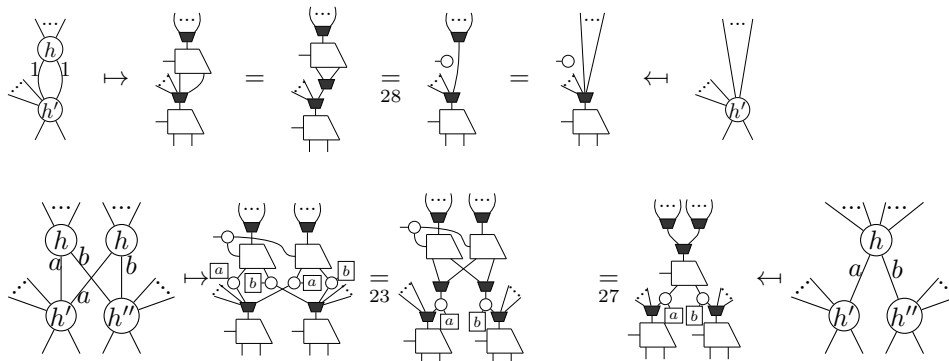
Proof. We start with the first rewrite of Figure 2, where $a \neq 0$:



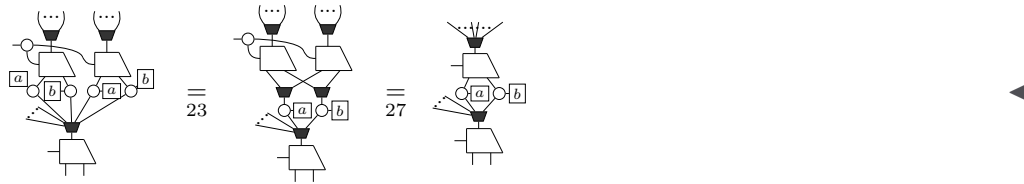
The proof for the rewrite where $a = 0$ and b diffuses instead is similar. When applied to the root, the proofs are the same with the additional equality $\overline{\bullet} \circ r \rightarrow r \bullet$.



The rule $\overline{\bullet} \circ r \rightarrow r \bullet$ is a direct consequence of the rewrite strategy of the proof of Proposition 20.





The case where the two children are the same is very similar:




Thanks to this proof, we now have a strategy to systematically reduce any ZH-diagram that is in SQMDD form. We then show how to put a ZH-diagram in SQMDD form in the first place.

5.2 SQMDD form of Generators and Compositions

We now show that all the generators of the ZH-Calculus can be put in SQMDD form, and furthermore that the compositions of diagrams in SQMDD form can be put in SQMDD form.

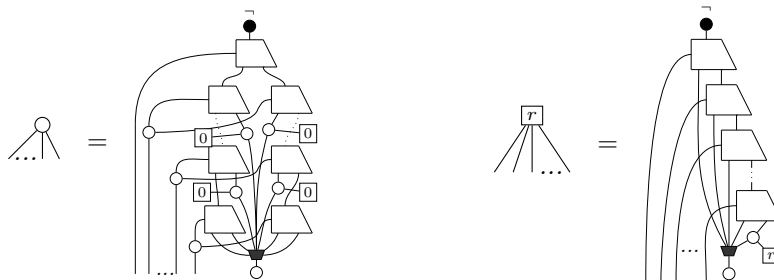
► **Proposition 30.** *Suppose D' differs from a diagram D in SQMDD form by replacing some of its  by .* We then have: $ZH \vdash D = D'$.

This result actually still holds for any commutative monoid whose neutral element is . A straightforward analysis yields that such monoids are of the form $\begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 1 & b \end{pmatrix}$ for $a, b \in \mathbb{C}$.

This result will be used in the proposition that follows, but notice that it can also be used to simplify the ZH-diagram obtained from an SQMDD.

Now, we can show that all the generators can be set in SQMDD form:

► **Proposition 31.** *The generators of the ZH-Calculus can be set in SQMDD form:*



We then need to show that the composition of diagrams in SQMDD form can be put in SQMDD form.

► **Proposition 32.** *The parallel composition of ZH-diagrams in SQMDD form can be put in SQMDD form, by joining the root of the right diagram to the terminal node of the diagram on the left.*

Since we work only with quantum states after using the map/state duality, what accounts for sequential composition is the linking of two ZH-diagrams in SQMDD form through \cup . This element can be decomposed into two primitives: $\cup = \begin{pmatrix} \cup \\ \circ \end{pmatrix}$. We hence show that applying each of these two primitives to a ZH-diagram in SQMDD form can be put in SQMDD form. To simplify the proof, we first show that swapping two outputs of a state in SQMDD can be put in SQMDD form.

► **Proposition 33.** *Let D be a ZH-diagram in SQMDD form. Applying \bowtie on two of its outputs can be put in SQMDD form. By repeated application of \bowtie , any permutation of the outputs of D can be put in SQMDD form.*

We can then move on to the two subproofs that show that applying \cup to a diagram in SQMDD form can be turned in SQMDD form:

► **Proposition 34.** *Let D be a ZH-diagram in SQMDD form. Applying \cup on two of its outputs can be put in SQMDD form.*

► **Proposition 35.** *Let D be a ZH-diagram in SQMDD form. Applying \circ on any of its outputs can be put in SQMDD form.*

Combining the previous propositions, we get the expected result:

► **Theorem 36.** *Any ZH-diagram can be put in reduced SQMDD form.*

This theorem gives an algorithm to translate any ZH-diagram into an SQMDD. Indeed, since a diagram D in reduced SQMDD form is in the image of $[\cdot]^{ZH}$ (which is injective), it now suffices to take its inverse to get a proper SQMDD.

References

- 1 Matthew Amy. Towards large-scale functional verification of universal quantum circuits. In Peter Selinger and Giulio Chiribella, editors, *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–21, 2019. doi:10.4204/EPTCS.287.1.
- 2 Miriam Backens and Aleks Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. In Peter Selinger and Giulio Chiribella, editors, *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 23–42, 2019. doi:10.4204/EPTCS.287.2.
- 3 Miriam Backens, Aleks Kissinger, Hector Miller-Bakewell, John van de Wetering, and Sal Wolfs. Completeness of the ZH-calculus, 2021.
- 4 Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. There and back again: A circuit extraction tale. *arXiv: Quantum Physics*, 2020. arXiv:2003.01664.
- 5 Lukas Burgholzer, Rudy Raymond, and Robert Wille. Verifying Results of the IBM Qiskit Quantum Circuit Compilation Flow, 2020.
- 6 Lukas Burgholzer and Robert Wille. Advanced Equivalence Checking for Quantum Circuits. *arXiv e-prints*, 2020. arXiv:2004.08420.
- 7 Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren, and Dominic Horsman. Graphical structures for design and verification of quantum error correction, 2016. arXiv:1611.08012.
- 8 Man-Duen Choi. Completely positive linear maps on complex matrices. *Linear Algebra and its Applications*, 10(3):285–290, 1975. doi:10.1016/0024-3795(75)90075-0.
- 9 Bob Coecke and Ross Duncan. Interacting quantum observables: Categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, April 2011. doi:10.1088/1367-2630/13/4/043016.
- 10 Bob Coecke and Aleks Kissinger. The compositional structure of multipartite quantum entanglement. In *Automata, Languages and Programming*, pages 297–308. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-14162-1_25.

- 11 Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Fast and effective techniques for T-count reduction via spider nest identities, 2020.
- 12 Niel de Beaudrap, Ross Duncan, Dominic Horsman, and Simon Perdrix. Pauli Fusion: a computational model to realise quantum transformations from ZX terms. In *QPL'19 : International Conference on Quantum Physics and Logic*, Los Angeles, United States, 2019. 12 pages + appendices. URL: <https://hal.archives-ouvertes.fr/hal-02413388>.
- 13 Niel de Beaudrap and Dominic Horsman. The ZX-calculus is a language for surface code lattice surgery. *CoRR*, abs/1704.08670, 2017. [arXiv:1704.08670](https://arxiv.org/abs/1704.08670).
- 14 Ross Duncan and Maxime Lucas. Verifying the Steane code with Quantomatic. In Bob Coecke and Matty Hoban, editors, *Proceedings of the 10th International Workshop on Quantum Physics and Logic, Castelldefels (Barcelona), Spain, 17th to 19th July 2013*, volume 171 of *Electronic Proceedings in Theoretical Computer Science*, pages 33–49. Open Publishing Association, 2014. doi:10.4204/EPTCS.171.4.
- 15 Ross Duncan and Simon Perdrix. Rewriting measurement-based quantum computations with generalised flow. *Lecture Notes in Computer Science*, 6199:285–296, 2010. doi:10.1007/978-3-642-14162-1_24.
- 16 Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. Two complete axiomatisations of pure-state qubit quantum computing. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18*, pages 502–511, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209128.
- 17 Anne Hillebrand. Quantum Protocols involving Multiparticle Entanglement and their Representations. Master's thesis, University of Oxford, 2011. URL: <https://www.cs.ox.ac.uk/people/bob.coecke/Anne.pdf>.
- 18 Andrzej Jamiołkowski. Linear transformations which preserve trace and positive semi-definiteness of operators. *Reports on Mathematical Physics*, 3(4):275–278, 1972. doi:10.1016/0034-4877(72)90011-0.
- 19 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18*, pages 559–568, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209131.
- 20 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A Generic Normal Form for ZX-Diagrams and Application to the Rational Angle Completeness. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, June 2019. doi:10.1109/LICS.2019.8785754.
- 21 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of the ZX-Calculus. *Logical Methods in Computer Science*, Volume 16, Issue 2, June 2020. doi:10.23638/LMCS-16(2:11)2020.
- 22 Aleks Kissinger and John van de Wetering. Reducing the number of non-Clifford gates in quantum circuits. *Phys. Rev. A*, 102:022406, August 2020. doi:10.1103/PhysRevA.102.022406.
- 23 Louis Lemonnier, John van de Wetering, and Aleks Kissinger. Hypergraph simplification: Linking the path-sum approach to the ZH-calculus, 2020. [arXiv:2003.13564](https://arxiv.org/abs/2003.13564).
- 24 D. M. Miller and M. A. Thornton. QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits. In *36th International Symposium on Multiple-Valued Logic (ISMVL'06)*, pages 30–30, 2006. doi:10.1109/ISMVL.2006.35.
- 25 P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler. QMDDs: Efficient Quantum Function Representation and Manipulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(1):86–99, 2016. doi:10.1109/TCAD.2015.2459034.
- 26 Philipp Niemann, Robert Wille, and Rolf Drechsler. Advanced exact synthesis of Clifford+T circuits. *Quantum Information Processing*, 19(9):317, 2020. doi:10.1007/s11128-020-02816-0.

- 27 Renaud Vilmart. A near-minimal axiomatisation of zx-calculus for pure qubit quantum mechanics. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, June 2019. doi:10.1109/LICS.2019.8785765.
- 28 Renaud Vilmart. The Structure of Sum-Over-Paths, its Consequences, and Completeness for Clifford, 2020. arXiv:2003.05678.
- 29 A. Zulehner, S. Hillmich, I. L. Markov, and R. Wille. Approximation of Quantum States Using Decision Diagrams. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 121–126, 2020. doi:10.1109/ASP-DAC47756.2020.9045454.

Decision Problems for Origin-Close Top-Down Tree Transducers

Sarah Winter 

Université libre de Bruxelles, Brussels, Belgium

Abstract

Tree transductions are binary relations of finite trees. For tree transductions defined by non-deterministic top-down tree transducers, inclusion, equivalence and synthesis problems are known to be undecidable. Adding origin semantics to tree transductions, i.e., tagging each output node with the input node it originates from, is a known way to recover decidability for inclusion and equivalence. The origin semantics is rather rigid, in this work, we introduce a similarity measure for transducers with origin semantics and show that we can decide inclusion, equivalence and synthesis problems for origin-close non-deterministic top-down tree transducers.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Transducers

Keywords and phrases tree transducers, equivalence, uniformization, synthesis, origin semantics

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.90

Related Version *Full Version*: <https://arxiv.org/abs/2107.02591>

Funding This research was supported by the DFG grant LO 1174/3.

Acknowledgements The author wants to thank Emmanuel Filiot for valuable feedback on the presentation.

1 Introduction

In this paper we study decision problems for top-down tree transducers over finite trees with origin semantics. Rounds [30] and Thatcher [31] independently invented tree transducers (their model is known today as top-down tree transducer) as a generalization of finite state word transducers in the context of natural language processing and compilers in the beginning of the 1970s. Nowadays, there is a rich landscape of various tree transducer models used in many fields, for example, syntax-directed translation [18], databases [29, 20], linguistics [27, 5], programming languages [33, 28], and security analysis [23].

Unlike tree automata, tree transducers have undecidable inclusion and equivalence problems [13]. This is already the case for word transducers [19, 17]. The intractability of, e.g., the equivalence problem for transducers (whether two given transducers recognize the same transduction, that is, the same relation) mainly stems from the fact that two transducers recognizing the same transduction may produce their outputs very differently. One transducer may produce its output fast and be ahead of the other. In general, there is an infinite number of transducers for a single transduction. To overcome this difficulty Bojanczyk [1] has introduced origin semantics, that is, additionally, there is an origin function that maps output positions to their originating input positions. The main result of [1] is a machine-independent characterization of transductions defined by deterministic two-way transducers with origin semantics. Word transducers with origin semantics were further investigated in [2], and properties of subclasses of transductions with origin semantics definable by one-way word transducers have been studied in [14, 9]. Under origin semantics, many interesting problems become decidable, e.g., equivalence of one-way word transducers. This is not surprising as a transduction now incorporates *how* it translates an input word into an output word providing much more information.



© Sarah Winter;

licensed under Creative Commons License CC-BY 4.0

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).

Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 90; pp. 90:1–90:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In [16], the authors have initiated a study of several decision problems for different tree transducer models on finite trees with origin semantics. More concretely, they studied inclusion, equivalence, injectivity and query determinacy problems for top-down tree transducers, tree transducers definable in monadic second order logic, and top-down tree-to-word transducers. They showed (amongst other results) that inclusion and equivalence become decidable for all models except tree-to-string transducers with origin semantics.

In general, there has been an interest to incorporate some kind of origin information (i.e., *how* a transduction works) into tree transductions, in order to gain more insight on different tree transductions, see, e.g., [32, 11, 26].

However, the origin semantics is rather rigid. To mitigate this, in [15], the authors have introduced a similarity measure between (one-way) word transducers with origin semantics which amounts to a measure that compares the difference between produced outputs on the same input prefix, in short, the measure compares their output delays. They show that inclusion, equivalence, and sequential uniformization (see next paragraph) problems become decidable for transducers that have bounded output delay. These problems are undecidable for word transducers in general, see [19, 17, 7]. The introduction of this similarity measure has triggered similar works on two-way word transducers, see [4, 3].

In order to obtain decidability results (in a less rigid setting than origin semantics), we initiate the study of inclusion, equivalence, and uniformization problems for top-down tree transducers under similarity measures which are based on the behavior of the transducers.

A uniformization of a binary relation is a function that selects for each element of the domain of the relation an element in its image. Synthesis problems are closely related to *effective* uniformization problems; algorithmic synthesis of specifications (i.e., relations) asks for effective uniformization by functions that can be implemented in a specific way. The classical setting is Church's synthesis problem [8], where logical specifications over infinite words are considered. Büchi and Landweber [6] showed that for specifications in monadic second order logic, that is, specifications that can be translated into synchronous finite automata, it is decidable whether they can be realized by a synchronous sequential transducer. Later, decidability has been extended to asynchronous sequential transducers [22, 21]. Detailed studies of the synthesis of sequential transducers from synchronous and asynchronous finite automata on finite words are provided in [15, 34], for an overview see [7].

Uniformization questions in this spirit have been first studied for relations over finite trees in [25, 24]. The authors have considered tree-automatic relations, that is, relations definable by tree automata over a product alphabet. They have shown that for tree-automatic relations definable by deterministic top-down tree automata uniformization by deterministic top-down tree transducers (which are a natural extension of sequential transducer on words) is decidable. However, for non-deterministic top-down tree automata it becomes undecidable.

Our contribution is the introduction of two similarity measures for top-down tree transducers. The first measure is an extension of the output delay measure introduced for word transducers in [15] to tree transducers. Comparing top-down tree transducers based on their output delay has also been done in e.g., [12], we use the same notion of delay to define our measure. Unfortunately, while decidability for major decision problems is regained in the setting of word transducers, we show that it is not in the setting of tree transducers. The second similarity measure is more closely connected to the origin semantics. We define two transducers as origin-close if there is a bound on the distance of two positions which are origins of the same output node by the two transducers. Our main result is that inclusion, equivalence and uniformization by deterministic top-down tree transducers is decidable for origin-close top-down tree transducers.

The paper is structured as follows. In Section 2 we provide definitions and terminology used throughout the paper. In Section 3 we present two similarity measures for (top-down tree) transducers and provide a comparison of their expressiveness, and in Section 4 we consider decision problems for origin-close top-down tree transducers.

2 Preliminaries

Words, trees, and contexts. An *alphabet* Σ is a finite non-empty set of *letters* or *symbols*. A finite *word* is a finite sequence of letters. The set of all finite words over Σ is denoted by Σ^* . The length of a word $w \in \Sigma^*$ is denoted by $|w|$, the empty word is denoted by ε . We write $u \sqsubseteq w$ if there is some v such that $w = uv$ for $u, v \in \Sigma^*$. A subset $L \subseteq \Sigma^*$ is called *language* over Σ . A *ranked alphabet* Σ is an alphabet where each letter $f \in \Sigma$ has a rank $rk(f) \in \mathbb{N}$. The set of letters of rank i is denoted by Σ_i . A tree domain dom is a non-empty finite subset of $(\mathbb{N} \setminus \{0\})^*$ such that dom is prefix-closed and for each $u \in (\mathbb{N} \setminus \{0\})^*$ and $i \in \mathbb{N} \setminus \{0\}$ if $ui \in dom$, then $uj \in dom$ for all $1 \leq j < i$. We speak of ui as successor of u for each $u \in dom$ and $i \in \mathbb{N} \setminus \{0\}$, and the \sqsubseteq -maximal elements of dom are called *leaves*.

A (finite Σ -labeled) *tree* is a mapping $t : dom_t \rightarrow \Sigma$ such that for each node $u \in dom_t$ the number of successors of u is a rank of $t(u)$. The height h of a tree t is the length of its longest path, i.e., $h(t) = \max\{|u| \mid u \in dom_t\}$. The set of all Σ -labeled trees is denoted by T_Σ . A subset $T \subseteq T_\Sigma$ is called *tree language* over Σ .

A *subtree* $t|_u$ of a tree t at node u is defined by $dom_{t|_u} = \{v \in \mathbb{N}^* \mid uv \in dom_t\}$ and $t|_u(v) = t(uv)$ for all $v \in dom_{t|_u}$. In order to formalize concatenation of trees, we introduce the notion of special trees. A *special tree* over Σ is a tree over $\Sigma \cup \{\circ\}$ such that \circ has rank zero and occurs exactly once at a leaf. Given $t \in T_\Sigma$ and $u \in dom_t$, we write $t[\circ/u]$ for the special tree that is obtained by deleting the subtree at u and replacing it by \circ . Let S_Σ be the set of special trees over Σ . For $t \in S_\Sigma$ and $s \in T_\Sigma$ or $s \in S_\Sigma$ let the *concatenation* $t \cdot s$ be the tree that is obtained from t by replacing \circ with s .

Let X_n be a set of n variables $\{x_1, \dots, x_n\}$ and Σ be a ranked alphabet. We denote by $T_\Sigma(X_n)$ the set of all trees over Σ which additionally can have variables from X_n at their leaves. We define X_0 to be the empty set, the set $T_\Sigma(\emptyset)$ is equal to T_Σ . Let $X = \bigcup_{n>0} X_n$. A tree from $T_\Sigma(X)$ is called *linear* if each variable occurs at most once. For $t \in T_\Sigma(X_n)$ let $t[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$ be the tree that is obtained by substituting each occurrence of $x_i \in X_n$ by $t_i \in T_\Sigma(X)$ for every $1 \leq i \leq n$.

A tree from $T_\Sigma(X_n)$ such that all variables from X_n occur exactly once and in the order x_1, \dots, x_n when reading the leaf nodes from left to right, is called *n-context* over Σ . Given an *n-context*, the node labeled by x_i is referred to as *i*th hole for every $1 \leq i \leq n$. A special tree can be seen as a 1-context, a tree without variables can be seen as a 0-context. If C is an *n-context* and $t_1, \dots, t_n \in T_\Sigma(X)$ we write $C[t_1, \dots, t_n]$ instead of $C[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$.

Tree transductions, origin mappings, and uniformizations. Let Σ, Γ be ranked alphabets. A *tree transduction* (from T_Σ to T_Γ) is a relation $R \subseteq T_\Sigma \times T_\Gamma$. Its *domain*, denoted $\text{dom}(R)$, is the projection of R on its first component. Given trees t_1, t_2 , an *origin mapping* of t_2 in t_1 is a function $o : dom_{t_2} \rightarrow dom_{t_1}$. Given $v \in dom_{t_2}$, $u \in dom_{t_1}$, we say v has origin u if $o(v) = u$. Examples are depicted in Figures 1g and 2. A *uniformization* of a tree transduction $R \subseteq T_\Sigma \times T_\Gamma$ is a function $f : \text{dom}(R) \rightarrow T_\Gamma$ such that $(t, f(t)) \in R$ for all $t \in \text{dom}(R)$.

Top-down tree transducers. We consider top-down tree transducers, which read the tree from the root to the leaves in a parallel fashion and produce finite output trees in each step that are attached to the already produced output.

A *top-down tree transducer* (a TDTT) is of the form $\mathcal{T} = (Q, \Sigma, \Gamma, q_0, \Delta)$ consisting of a finite set of states Q , a finite input alphabet Σ , a finite output alphabet Γ , an initial state $q_0 \in Q$, and Δ is a finite set of transition rules of the form

$$q(f(x_1, \dots, x_i)) \rightarrow w[q_1(x_{j_1}), \dots, q_n(x_{j_n})],$$

where $f \in \Sigma_i$, w is an n -context over Γ , $q, q_1, \dots, q_n \in Q$ and variables $x_{j_1}, \dots, x_{j_n} \in X_i$. A *deterministic* TDTT (a DTDTT) has no two rules with the same left-hand side.

We now introduce a non-standard notion of configurations which is more suitable to prove our results. Usually, a configuration is a partially transformed input tree; the upper part is the already produced output, the lower parts are remainders of the input tree. Here, we keep the input and output tree separate and introduce a mapping from nodes of the output tree to nodes of the input tree from where the transducer continues to read. A visualization of several configurations is given in Figure 1.

A *configuration* of a top-down tree transducer is a triple $c = (t, t', \varphi)$ of an input tree $t \in T_\Sigma$, an output tree $t' \in T_{\Gamma \cup Q}$ and a function $\varphi : D_{t'} \rightarrow \text{dom}_t$, where

- $t'(u) \in \Gamma_i$ for each $u \in \text{dom}_{t'}$ with $i > 0$ successors, and
- $t'(u) \in \Gamma_0$ or $t'(u) \in Q$ for each leaf $u \in \text{dom}_{t'}$, and
- $D_{t'} \subseteq \text{dom}_{t'}$ with $D_{t'} = \{u \in \text{dom}_{t'} \mid t'(u) \in Q\}$, i.e., φ maps every node from the output tree t' that has a state-label to a node of the input tree t .

Let $c_1 = (t, t_1, \varphi_1)$ and $c_2 = (t, t_2, \varphi_2)$ be configurations of a top-down tree transducer over the same input tree. We define a *successor relation* $\rightarrow_{\mathcal{T}}$ on configurations as usual by applying one rule. Figure 1 illustrates a configuration sequence explained in Example 1 below. Formally, for the application of a rule, we define $c_1 \rightarrow_{\mathcal{T}} c_2 \Leftrightarrow$

- There is a state-labeled node $u \in D_{t'}$ of the output tree t_1 that is mapped to a node $v \in \text{dom}_t$ of the input tree t , i.e., $\varphi_1(u) = v$, and
- there is a rule $t_1(u) (t(v)(x_1, \dots, x_i)) \rightarrow w[q_1(x_{j_1}), \dots, q_n(x_{j_n})] \in \Delta$ such that the output tree is correctly updated, i.e., $t_2 = t_1[\circ/u] \cdot w[q_1, \dots, q_n]$, and
- the mapping φ_2 is correctly updated, i.e., $\varphi_2(u') = \varphi_1(u')$ if $u' \in D_{t_1} \setminus \{u\}$ and $\varphi_2(u') = v.j_i$ if $u' = u.u_i$ with u_i is the i th hole in w .

Furthermore, let $\rightarrow_{\mathcal{T}}^*$ be the reflexive and transitive closure of $\rightarrow_{\mathcal{T}}$. From here on, let φ_0 always denote the mapping $\varphi_0(\varepsilon) = \varepsilon$. A configuration (t, q_0, φ_0) is called *initial configuration* of \mathcal{T} on t . A configuration sequence starting with an initial configuration where each configuration is a successor of the previous one is called a *run*. For a tree $t \in T_\Sigma$ let $\mathcal{T}(t) \subseteq T_{\Gamma \cup Q}$ be the set of *final transformed outputs* of a computation of \mathcal{T} on t , that is the set $\{t' \mid (t, q_0, \varphi_0) \rightarrow_{\mathcal{T}}^* (t, t', \varphi) \text{ s.t. there is no successor configuration of } (t, t', \varphi)\}$. Note, we explicitly do not require that the final transformed output is a tree over Γ . In the special case that $\mathcal{T}(t)$ is a singleton set $\{t'\}$, we also write $\mathcal{T}(t) = t'$. The *transduction* $R(\mathcal{T})$ induced by a TDTT \mathcal{T} is $R(\mathcal{T}) = \{(t, t') \mid t' \in \mathcal{T}(t) \cap T_\Gamma\}$. The class of relations definable by TDTTs is called the class of *top-down tree transductions*, conveniently denoted by TDTT.

Let \mathcal{T} be a TDTT, and let $\rho = c_0 \dots c_n$ be a run of \mathcal{T} on an input tree $t \in T_\Sigma$ that results in an output tree $s \in T_\Gamma$. The *origin function* o of ρ maps a node u of the output tree to the node v of the input tree that was read while producing u , formally $o : \text{dom}_s \rightarrow \text{dom}_t$ with $o(u) = v$ if there is some i , such that $c_i = (t, t_i, \varphi_i)$, $c_{i+1} = (t, t_{i+1}, \varphi_{i+1})$ and $\varphi_i(u) = v$ and $t_{i+1}(u) = s(u)$, see Figure 1. We define $R_o(\mathcal{T})$ to be the set

$$\{(t, s, o) \mid t \in T_\Sigma, s \in T_\Gamma \text{ and } \exists \rho : (t, q_0, \varphi) \rightarrow_{\mathcal{T}}^* (t, s, \varphi') \text{ with origin } o\}.$$

► **Example 1.** Let Σ be a ranked alphabet given by $\Sigma_2 = \{f\}$, $\Sigma_1 = \{g, h\}$, and $\Sigma_0 = \{a\}$. Consider the TDTT \mathcal{T} given by $(\{q\}, \Sigma, \Sigma, \{q\}, \Delta)$ with $\Delta = \{q(a) \rightarrow a, q(g(x_1)) \rightarrow q(x_1), q(h(x_1)) \rightarrow h(q(x_1)), q(f(x_1, x_2)) \rightarrow f(q(x_1), q(x_2))\}$. For each $t \in T_\Sigma$ the transducer deletes

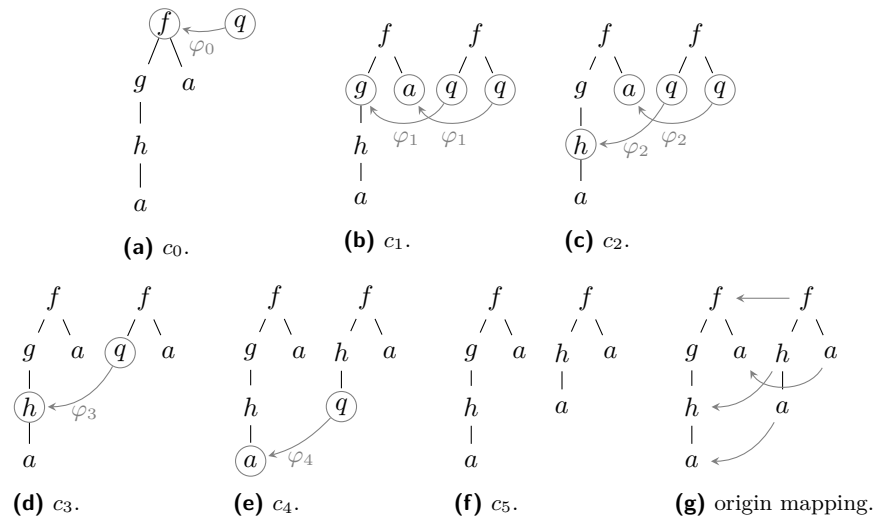


Figure 1 The configuration sequence c_0 to c_5 of \mathcal{T} on $f(g(h(a)), a)$ and resulting origin mapping from Example 1.

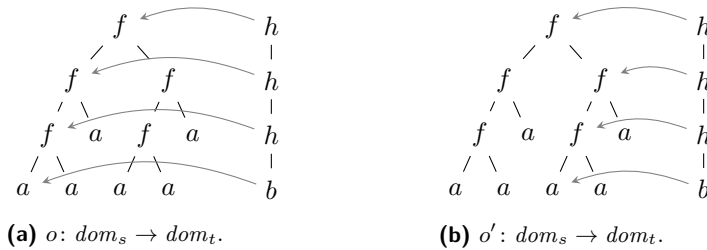


Figure 2 Origin mappings o, o' . We have that $dist(o(111), o'(111))$, that is, the distance of the origins of the leaf node, is the length of the shortest path from node 111 to node 211 which is 6.

all occurrences of g in t . Consider $t := f(g(h(a)), a)$. A possible sequence of configurations of \mathcal{T} on t is $c_0 \rightarrow^5 c_5$ such that $c_0 := (t, q, \varphi_0)$ with $\varphi_0(\varepsilon) = \varepsilon$, $c_1 := (t, f(q, q), \varphi_1)$ with $\varphi_1(1) = 1$, $\varphi_1(2) = 2$, $c_2 := (t, f(q, q), \varphi_2)$ with $\varphi_2(1) = 11$, $\varphi_2(2) = 2$, $c_3 := (t, f(q, a), \varphi_3)$ with $\varphi_3(1) = 11$, $c_4 := (t, f(h(q), a), \varphi_4)$ with $\varphi_4(11) = 111$, and $c_5 := (t, f(h(a), a), \varphi_5)$. A visualization of this sequence and resulting origin mapping is shown in Figure 1.

► **Example 2.** Let Σ, Γ be given by $\Sigma_2 = \{f\}$, $\Sigma_0 = \{a\}$, $\Gamma_1 = \{h\}$, and $\Gamma_0 = \{b\}$. Consider the TDTT \mathcal{T} given by $(\{q\}, \Sigma, \Gamma, \{q\}, \Delta)$ with $\Delta = \{ q(a) \rightarrow b, q(f(x_1, x_2)) \rightarrow h(q(x_1)), q(f(x_1, x_2)) \rightarrow h(q(x_2)) \}$. Basically, when reading an f -labeled node, the TDTT can non-deterministically decide whether to continue reading in left or the right subtree. In Figure 2 two origin mappings $o: dom_s \rightarrow dom_t$ and $o': dom_s \rightarrow dom_t$ are given that are result of runs of \mathcal{T} on $t = f(f(f(a, a), a), a), f(f(a, a), a), a)$ with final output $s = h(h(h(b)))$.

In this work, we focus on decision problems for transducers with origin semantics. To begin with, we introduce some notations and state relevant known results in this context.

Shorthand notations. Let \mathcal{C} denote a class of transducers with origin semantics, e.g., TDTT or DTDTT. Given a class \mathcal{C} and $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, if $R(\mathcal{T}_1) \subseteq R(\mathcal{T}_2)$ (resp. $R_o(\mathcal{T}_1) \subseteq R_o(\mathcal{T}_2)$), we write $\mathcal{T}_1 \subseteq \mathcal{T}_2$ (resp. $\mathcal{T}_1 \subseteq_o \mathcal{T}_2$). Furthermore, if $R(\mathcal{T}_1) = R(\mathcal{T}_2)$ (resp. $R_o(\mathcal{T}_1) = R_o(\mathcal{T}_2)$),

we write $\mathcal{T}_1 = \mathcal{T}_2$ (resp. $\mathcal{T}_1 =_o \mathcal{T}_2$). Given classes $\mathcal{C}_1, \mathcal{C}_2$, $\mathcal{T}_1 \in \mathcal{C}_1$, and $\mathcal{T}_2 \in \mathcal{C}_2$, if \mathcal{T}_1 defines a function f that is a uniformization of $R(\mathcal{T}_2)$, we say \mathcal{T}_1 uniformizes \mathcal{T}_2 , if additionally $\mathcal{T}_1 \subseteq_o \mathcal{T}_2$, we say \mathcal{T}_1 origin uniformizes \mathcal{T}_2 .

Decision problems. The *inclusion* resp. *origin inclusion problem* for a class \mathcal{C} asks, given $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, whether $\mathcal{T}_1 \subseteq \mathcal{T}_2$ resp. $\mathcal{T}_1 \subseteq_o \mathcal{T}_2$. The *equivalence* resp. *origin equivalence problem* for a class \mathcal{C} asks, given $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, whether $\mathcal{T}_1 = \mathcal{T}_2$ resp. $\mathcal{T}_1 =_o \mathcal{T}_2$. Lastly, the *uniformization* resp. *origin uniformization problem* for classes $\mathcal{C}_1, \mathcal{C}_2$ asks, given $\mathcal{T}_2 \in \mathcal{C}_2$, whether there exists $\mathcal{T}_1 \in \mathcal{C}_1$ such that \mathcal{T}_1 uniformizes (resp. origin uniformizes) \mathcal{T}_2 .

As mentioned in the introduction, generally, a transduction can be defined by several transducers behaving very differently, making many problems intractable. Adding origin semantics to transducers, i.e., seeing the transducer behavior as part of the transduction, allows to recover decidability. The following is known for the class TDDT.

► **Theorem 3** ([13]). *Inclusion and equivalence are undecidable for the class TDDT.*

► **Theorem 4** ([16]). *Origin inclusion and origin equivalence are decidable for the class TDDT.*

Turning to uniformization problems, it is known that every TDDT is uniformizable by a DTDDT with regular lookahead (a DTDDT^R), that is, the transducer can check membership of the subtrees of a node in regular tree-languages before processing the node.

► **Theorem 5** ([10]). *Every TDDT has a DTDDT^R-uniformization.*

However, when requiring that the input should be transformed on-the-fly (without regular lookahead), the uniformization problem becomes undecidable. In [7], it was shown that it is undecidable whether a one-way (non-deterministic) word transducer has a uniformization by a sequential transducer (that is, basically, a one-way deterministic transducer). So, we get undecidability in the tree setting for free (as stated in Theorem 6). This problem has not been investigated with origin semantics so far. We show decidability (also for more relaxed versions), see Theorem 16.

► **Theorem 6.** *DTDDT^R-uniformization is undecidable for the class TDDT.*

Since the origin semantics is rather rigid, in the next section, we introduce two similarity measures between transducers which are based on their behavior and re-investigate the introduced decision problems for transducers with “similar” behavior.

3 Similarity measures for transducers

An idea that naturally comes to mind is to say that two transducers behave similarly if for two computations over the same input that yield the same output their respective origin mappings are “similar”.

The other idea is to say that two computations are similar if their output delay is small, roughly meaning that for the same prefix (for an adequate notion of prefix for trees) of the input the so-far produced output is of similar size. Decision problems using this measure have already been investigated for (one-way) word transducers [15], we lift the measure to top-down tree transducers.

Origin distance. Given a tree t , let the distance between two nodes $u, v \in \text{dom}_t$, written $\text{dist}(u, v)$, be the shortest path between u and v (ignoring the edge directions), an example is given in Figure 2.

Given $\mathcal{T}, \mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, where \mathcal{C} is a class of transducers with origin semantics. We say (t, s, o) is *k-origin included* in $R_o(\mathcal{T})$, written $(t, s, o) \in_k R_o(\mathcal{T})$, if there is $(t, s, o') \in R_o(\mathcal{T})$ such that $\text{dist}(o(i), o'(i)) \leq k$ for all $i \in \text{dom}_s$. We say \mathcal{T}_1 is *k-origin included* in \mathcal{T}_2 , written $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$, if $(s, t, o) \in_k R_o(\mathcal{T}_2)$ for all $(s, t, o) \in R_o(\mathcal{T}_1)$. We say \mathcal{T}_1 and \mathcal{T}_2 are *k-origin equivalent*, written $\mathcal{T}_1 =_k \mathcal{T}_2$, if $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$ and $\mathcal{T}_2 \subseteq_k \mathcal{T}_1$. We say \mathcal{T}_1 *k-origin uniformizes* \mathcal{T}_2 if $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$ and \mathcal{T}_1 uniformizes \mathcal{T}_2 . The *k-origin* decision problems are defined as expected.

We need some additional notations, before we can introduce the concept of delay.

Partial and prefix trees. Let N_Σ be the set of all trees over Σ which can have symbols from Σ , that is, symbols with rank ≥ 0 , at their leaves. The set N_Σ is the set of all *partial trees* over Σ . Note that N_Σ includes T_Σ . We say a tree $t' \in N_\Sigma$ is a *prefix tree* of a tree $t \in N_\Sigma$, written $t' \sqsubseteq t$, if $\text{dom}_{t'} \subseteq \text{dom}_t$, and $t'(u) = t(u)$ for all $u \in \text{dom}_{t'}$. Given $t_1, t_2 \in N_\Sigma$, its *greatest common prefix*, written $t_1 \wedge t_2$, is the tree $t \in N_\Sigma$ such that dom_t is the largest subset of $\text{dom}_{t_1} \cap \text{dom}_{t_2}$ such that $t \sqsubseteq t_1$ and $t \sqsubseteq t_2$. Removing $t_1 \wedge t_2$ from t_1 and t_2 naturally yields a set of partial trees (we omit a formal definition) called *difference trees*. These notions are visualized in Figure 3.

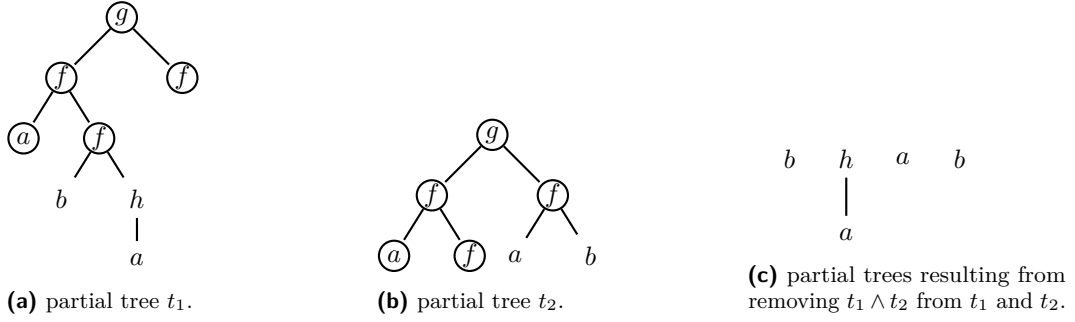
Delay. Given words w_1, w_2 , to compute their delay, we remove their greatest common prefix $w = w_1 \wedge w_2$, say $w_1 = wv_1$ and $w_2 = wv_2$, and their delay is the maximum of the length of their respective reminders, i.e., $\max\{|v_1|, |v_2|\}$. We lift this to trees, given (partial) trees t_1, t_2 , we remove their greatest common prefix $t_1 \wedge t_2$ from t_1 and t_2 which yields a set S of partial trees, we define their delay as $\text{delay}(t_1, t_2) = \max\{h(t) + 1 \mid t \in S\}$. An example is given in Figure 3. Note that for trees over unary and leaf symbols (a way to see words) the definitions for words and trees are equal. Recall that the length of the word a is one, but the height of the tree a is zero.

In order to define a similarity measure between transducers using delay, we take two transducer runs on the same input and compute the delay between their produced outputs throughout their runs. Although we have defined delay between words and trees, we only provide a formal definition for top-down tree transducers. However, for word transducers, examples are given in Example 17, and a formal definition can be found in [15].

Now, let \mathcal{T}_1 and \mathcal{T}_2 be TDTTs, and ρ_1 and ρ_2 be runs of \mathcal{T}_1 and \mathcal{T}_2 , respectively, over the same input tree $t \in T_\Sigma$ such that $\rho_1: (t, q_0^{\mathcal{T}_1}, \varphi_0) \rightarrow_{\mathcal{T}_1}^* (t, t_1, \varphi)$ with $t_1 \in T_\Gamma$, and $\rho_2: (t, q_0^{\mathcal{T}_2}, \varphi_0) \rightarrow_{\mathcal{T}_2}^* (t, t_2, \varphi')$ with $t_2 \in T_\Gamma$.

Basically, we take a look at all configurations that occur in the runs and compute the delay between the output trees of compatible configurations where compatible means in both configurations the same prefix (level-wise, see below) of the input tree has been processed.

Let us be a bit more clear what we mean with compatible. Note that when comparing two configuration sequences (i.e., runs) of word transducers the notion of “have processed the same input so far” is clear. For tree transducers, in one configuration sequence, a left-hand subtree might be processed before the right-hand subtree, and in another configuration sequence vice versa. Since these computation steps are done in a parallel fashion (just written down in an arbitrary order in the configuration sequence), we need to make sure to compare configurations where the subtrees have been processed equally far (we call this level-wise). Also, a tree transducer might not even read the whole input tree, as, e.g., in Example 2. We also (implicitly) take care of this in our definition.



■ **Figure 3** The greatest common prefix of the partial trees t_1 and t_2 , $t_1 \wedge t_2$, is marked with circles in t_1 and t_2 . The delay between t_1 and t_2 is computed from their non-common parts as $\text{delay}(t_1, t_2) = \max\{h(t) + 1 \mid t \in \{b, h(a), a\}\} = 2$.

The result is the maximum of the delay between output trees of compatible configurations. Given $t \in T_\Sigma$, let $\text{Pref}_{\text{level}}(t)$ denote the set of all prefix trees of t such that if a node at level i is kept, then all other nodes at level i are kept, i.e., for $t = f(h(a), h(a))$, $\text{Pref}_{\text{level}}(t)$ contains $f(h, h)$, but not $f(h(a), h)$. Given an intermediate configuration (t, t'_i, φ') of the run ρ_i , we recall that $t'_i \in T_{\Gamma \cup Q_{\mathcal{T}_i}}$ meaning t'_i contains states of \mathcal{T}_i as leaves. Let $t'_i|_\Gamma$ denote the partial tree obtained from t'_i by removing all non- Γ -labelled nodes. We define $\text{delay}(\rho_1, \rho_2)$ as

$$\max\{\text{delay}(t'_1|_\Gamma, t'_2|_\Gamma) \mid \text{there is } t' \in \text{Pref}_{\text{level}}(t), \text{ there is } (t, t'_1, \varphi') \text{ in } \rho_1 \text{ with } t'_1 \in \mathcal{T}_1(t'), \\ \text{and, there is } (t, t'_2, \varphi'') \text{ in } \rho_2 \text{ with } t'_2 \in \mathcal{T}_2(t')\}.$$

The conditions $t'_1 \in \mathcal{T}_1(t')$ and $t'_2 \in \mathcal{T}_2(t')$ are introduced to make sure that all input nodes that can be processed from t' are processed in the selected configurations.

We introduce (shorthand) notations. Let $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, where \mathcal{C} is a class of transducers. Given $(t, s) \in R(\mathcal{T}_1)$, we say (t, s) is k -delay included in $R(\mathcal{T}_2)$, written $(t, s) \in_{\mathbb{D}_k} R(\mathcal{T}_2)$, if there are runs ρ and ρ' of \mathcal{T}_1 and \mathcal{T}_2 , respectively, with input t and output s such that $\text{delay}(\rho, \rho') \leq k$. We say \mathcal{T}_1 is k -delay included in \mathcal{T}_2 , written $\mathcal{T}_1 \subseteq_{\mathbb{D}_k} \mathcal{T}_2$, if $(t, s) \in_{\mathbb{D}_k} R(\mathcal{T}_2)$ for all $(t, s) \in R(\mathcal{T}_1)$. We say \mathcal{T}_1 and \mathcal{T}_2 are k -delay equivalent, written $\mathcal{T}_1 =_{\mathbb{D}_k} \mathcal{T}_2$, if $\mathcal{T}_1 \subseteq_{\mathbb{D}_k} \mathcal{T}_2$ and $\mathcal{T}_2 \subseteq_{\mathbb{D}_k} \mathcal{T}_1$. We say \mathcal{T}_1 k -delay uniformizes \mathcal{T}_2 if $\mathcal{T}_1 \subseteq_{\mathbb{D}_k} \mathcal{T}_2$ and \mathcal{T}_1 uniformizes \mathcal{T}_2 . The k -delay decision problems are defined as expected.

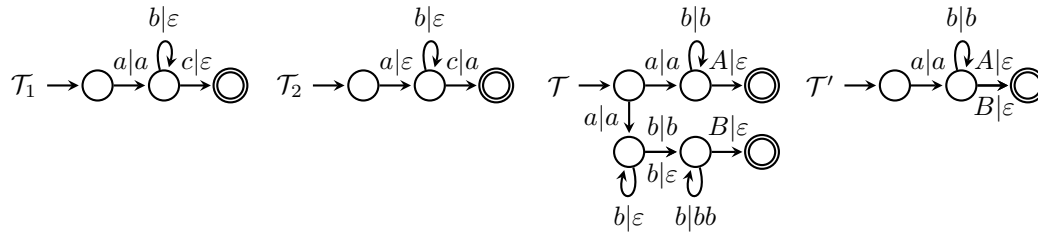
In order to get a better understanding of the expressiveness and differences between the two similarity measures, we first explore their properties on word transductions since words are a particular case of trees (i.e., monadic trees).

Word transducers. We denote by FST a *finite state transducer*, the class of word transductions recognized by FSTs is the class of *rational transductions*, conveniently also denoted by FST. We omit a formal definition of FSTs, because they are not considered outside of this section. An FST is *sequential* if its transitions are input-deterministic, more formally, a DTDTT over ranked alphabets with only unary and leaf symbols can be seen as an FST.

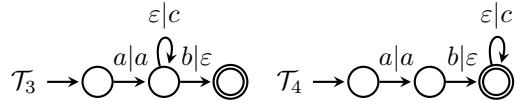
The results below concern origin distance, the same results were proven for delay in [15].

► **Proposition 7.**

1. There exist FSTs $\mathcal{T}_1, \mathcal{T}_2$ such that $\mathcal{T}_1 \subseteq \mathcal{T}_2$, but $\mathcal{T}_1 \not\subseteq_k \mathcal{T}_2$ for all $k \geq 0$.
2. There exist FSTs $\mathcal{T}_1, \mathcal{T}_2$ such that $\mathcal{T}_1 = \mathcal{T}_2$, but $\mathcal{T}_1 \not\equiv_k \mathcal{T}_2$ for all $k \geq 0$.
3. There exists an FST \mathcal{T} such that \mathcal{T} is sequentially uniformizable, but \mathcal{T} is not k -origin sequentially uniformizable for all $k \geq 0$.



(a) We have $\mathcal{T}_1 = \mathcal{T}_2$, and $\mathcal{T}_1 =_{\mathbb{D}_1} \mathcal{T}_2$, but $\mathcal{T}_1 \neq_k \mathcal{T}_2$ for all $k \geq 0$; \mathcal{T}' is a sequential uniformizer of \mathcal{T} .



(b) We have $\mathcal{T}_3 = \mathcal{T}_4$, and $\mathcal{T}_3 =_1 \mathcal{T}_4$, but $\mathcal{T}_3 \neq_{\mathbb{D}_k} \mathcal{T}_4$ for all $k \geq 0$.

■ **Figure 4** Comparing origin distance and delay for word transducers, see the proof of Proposition 7 and Example 17.

Proof. First, consider the FSTs $\mathcal{T}_1, \mathcal{T}_2$ depicted in Figure 4a. Both recognize the same function $f: \{a, b, c\}^* \rightarrow \{a\}^*$ defined as $f(ab^*c) = a$. Clearly, $\mathcal{T}_1 \subseteq \mathcal{T}_2$ and $\mathcal{T}_1 = \mathcal{T}_2$. However, the origin distance between \mathcal{T}_1 and \mathcal{T}_2 is unbounded. In \mathcal{T}_1 , the origin of the single output letter a is always the first input letter. In contrast, in \mathcal{T}_2 , the origin of the output letter a is always the last input letter. Secondly, consider the FSTs $\mathcal{T}, \mathcal{T}'$ depicted in Figure 4a. The recognized relation $\mathcal{R}(\mathcal{T}) \subseteq \{a, b, A, B\}^* \times \{a, b\}^*$ consists of $\{(ab^n A, ab^n) \mid n \in \mathbb{N}\}$ and $\{(ab^n B, ab^m) \mid 0 \leq m \leq 2n - 1, n \in \mathbb{N}\}$. The sequential transducer \mathcal{T}' recognizes the function $f: \{a, b, A, B\}^* \rightarrow \{a, b\}^*$ defined by $f(ab^n X) = ab^n$ for $X \in \{A, B\}$ and all $n \in \mathbb{N}$. Clearly, \mathcal{T}' is a sequential uniformization of \mathcal{T} . However, no sequential uniformization with bounded origin distance exists, see Appendix B. ◀

We give an example (depicted in Figure 4 and described in detail in Example 17) that shows that the two notions are orthogonal to each other. However, if we restrict the class FST to *real-time*¹ FST, that is, word transducers such that in every transition exactly one input symbol is read, the notion of delay is more powerful than origin distance, see below. It is important to note that we have proven Proposition 7 for real-time FSTs which are equivalent to TDFTs on monadic trees, i.e., Proposition 7 is true for the class TDFT.

► **Proposition 8.** *Let $\mathcal{T}_1, \mathcal{T}_2$ be real-time FSTs, if $\mathcal{T}_1 \subseteq_i \mathcal{T}_2$ for some $i \geq 0$, then $\mathcal{T}_1 \subseteq_{\mathbb{D}_j} \mathcal{T}_2$ for some $j \geq 0$.*

The notion of bounded delay is suitable to regain decidability.

► **Theorem 9** ([15]). *Given $k \geq 0$, k -delay inclusion, k -delay equivalence and k -delay sequential uniformization are decidable for the class FST.*

Ideally, we would like to lift Theorem 9 from word to tree transducers, but it turns out that the notion of delay is too expressive to yield decidability results for tree transducers as shown in the next paragraph.

¹ ε -transitions (as, e.g., the loop in \mathcal{T}_3 from Figure 4, not be confused with non-producing transitions) are standard for FST, and non-standard for TDFT in the literature. We consider “real-time” TDFT by default.

Tree transducers. It is undecidable whether a given tree-automatic relation has a uniformization by a synchronous DTDTT [24]. A DTDTT is called synchronous if for one processed input node one output node is produced as, e.g., in Example 2. Tree-automatic relations are a subclass of the relations that are recognizable by synchronous TDTT. To prove the result, the authors showed that

► **Lemma 10** ([24]). *There exists a synchronous TDTT \mathcal{T}_M , based on a Turing machine M , that is 0-delay DTDTT-uniformizable iff M halts on the empty input.*

In the proof, for a TM M , a DTDTT \mathcal{T}'_M is constructed such that \mathcal{T}'_M 0-delay uniformizes \mathcal{T}_M iff M halts on the empty input. Recall that this implies that $\mathcal{T}'_M \subseteq_{\mathbb{D}_0} \mathcal{T}_M$ iff M halts on the empty input. Consequently, we obtain that

► **Theorem 11.** *Given $k \geq 0$, k -delay inclusion and k -delay DTDTT-uniformization are undecidable for the class TDTT (even for $k = 0$).*

We do not know whether k -equivalence is decidable for a given $k \geq 0$. Note that Theorem 11 does not imply that 0-origin inclusion and 0-origin DTDTT-uniformization is undecidable for the class TDTT. For the class FST, the notions of 0-origin and 0-delay fall together, but for TDTT this is no longer the case. Recall the TDTT given in Example 2 and its unique runs that yield the origin mappings depicted in Figure 2. The delay between these runs is zero, but their origin mappings are different. An analysis of the (un)decidability proof(s) in [24] pins the problems down to the fact that in the specification and in the possible implementations the origins for the same output node lie on different paths in the input tree. For trees, this fact has no influence when measuring the delay between computations (as seen in Example 2). However, it is recognizable using the origin distance as measure. Since the notion of delay is so powerful that the decision problems under bounded delay become undecidable for tree transducers (see Theorem 11) in contrast to word transducers (see Theorem 9), in the next section, we focus on bounded origin distance.

4 Decision problems for origin-close transducers

We show that the decision problems become decidable for top-down tree transducers with bounded origin distance, see Theorem 16. The next part is devoted to explaining our proof ideas and introducing our main technical lemma (Lemma 13) which is used in all proofs.

Origin-close transductions are representable as regular tree languages. Given $k \geq 0$, and a TDTT \mathcal{T} , we construct an infinite tree $H_{\mathcal{T},k}$, given as the unfolding of a finite graph $G_{\mathcal{T},k}$, such that a node in this infinite tree represents an input sequence from a finite input tree and an output sequence (where the intuition is that this output sequence was produced while processing this input sequence). The idea is that in $H_{\mathcal{T},k}$, we define choices (aka. strategies) of two so-called players In and Out, where a strategy t of In together with a strategy s of Out defines an input tree t , an output tree s , and an origin mapping $o: dom_s \rightarrow dom_t$ of s in t . We can annotate the tree $H_{\mathcal{T},k}$ with the strategies t and s which yields a tree $H_{\mathcal{T},k} \hat{\curvearrowright} t \hat{\curvearrowright} s$. We use this game-like view for all considered decision problems. We illustrate this view.

► **Example 12.** Recall the TDTT \mathcal{T} over Σ and Γ given in Example 2. First, we explain how the graph $G_{\mathcal{T},0}$ looks like. Its unfolding is the infinite tree $H_{\mathcal{T},0}$ (with annotations t and s) depicted in Figure 5. We have three types of nodes: $\{\varepsilon, 1, 2\}$ to indicate that the current node is the root, a first or a second child. The maximum rank of Σ is two, hence $\{\varepsilon, 1, 2\}$. These nodes belong to In who can choose the input label, represented by nodes $\{f, a\}$. Then

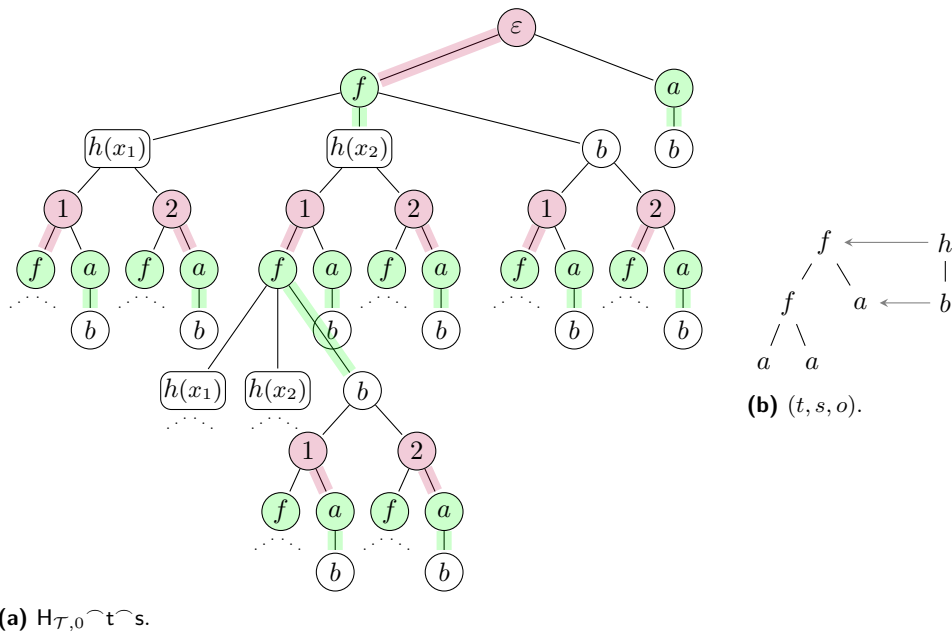


Figure 5 Infinite tree $H_{\mathcal{T},0}$ based on \mathcal{T} from Example 2. On red nodes In must make a choice, on green nodes Out must make a choice. Their respective strategies t and s which define their choices are highlighted on the edges in red and green, respectively. Together, t and s encode the input tree $t = f(f(a, a), a)$, the output tree $s = h(b)$ and origin mapping $o: dom_s \rightarrow dom_t$ as depicted. Note that since t and s are strategies, choices are made whatever the other player does, that is why in $H_{\mathcal{T},0} \hat{\sim} t \hat{\sim} s$, we also have, e.g., a green annotation at node 1112 even though In picked node 1111.

Out chooses which output (from $T_{\Gamma}(X)$) should be produced while processing a node. Since $k = 0$, and all right-hand sides of rules in \mathcal{T} have height at most one, only outputs of height at most one are suitable to maintain origin distance $k = 0$. For input f possible choices are $h(x_1)$ and $h(x_2)$, indicating whether to continue to process the left or the right subtree, or b . For input a only output b is possible. After the output, edges to $\{1, \dots, rk(\sigma)\}$ exist, where σ is the last seen input letter. Further explanation is given in Figure 5.

We present our main technical lemma which states that origin-close transductions are representable as tree language recognizable by a parity tree automaton (a PTA).

► **Lemma 13.** *Given $k \geq 0$ and a TDDT \mathcal{T} , there exists a PTA that recognizes the tree language $\{H_{\mathcal{T},k} \hat{\sim} t \hat{\sim} s \mid (t, s, o) \in_k R_o(\mathcal{T})\}$.*

Proof sketch. The infinite tree $H_{\mathcal{T},k} \hat{\sim} t \hat{\sim} s$ encodes a triple (t, s, o) . We construct a PTA (which has in fact a safety acceptance condition) that guesses a run of \mathcal{T} over the input tree t with output tree s that yields an origin mapping o' such that $(t, s, o') \in R_o(\mathcal{T})$ and $dist(o(i), o'(i)) \leq k$ which implies that $(t, s, o) \in_k R_o(\mathcal{T})$.

Checking whether $dist(o(i), o'(i)) \leq k$ can be done on-the-fly because the origin distance is bounded which implies that the difference trees of so-far produced output by the guessed run and the productions encoded by the annotations are of bounded size. Thus, they can be stored in the state space of the PTA. Even though the construction idea is rather simple, the implementation and correctness proof are non-trivial. We face two difficulties. Firstly, we have to account for the fact that in o and o' origins for the same output node can lie on different paths of the input tree. However, since their distance is bounded, the

amount of shared information that the PTA has to check on different paths is also bounded. Secondly, it is possible to have non-linear transformation rules (that is, rules with copy, e.g., $q(f(x_1, x_2)) \rightarrow f(q_1(x_2), q_2(x_2))$) which adds another layer of complication. This causes that an unbounded number of output nodes can have the same input node as origin. We require that $H_{\mathcal{T},k} \hat{\tau} \hat{s}$ is a tree over a ranked alphabet, hence we have to bound the number of output choices that can be made at an input node. We show that it suffices to only make a bounded number of output choices for each input node. The main insight is that when two continuations of the output tree depend on the same continuation of the input tree, then it suffices to only consider one of them (because the other one can be continued in the same way) if they share the same relevant information where relevant basically means that the state that \mathcal{T} has reached (guessed by the PTA) at these two output nodes and the output difference trees compared to Out 's choices (given by s) are the same. ◀

Solving decision problems for origin-close transducers. We show that deciding k -origin inclusion and equivalence for TDTTs reduces to deciding language inclusion for PTAs.

► **Proposition 14.** *Given $k \geq 0$, k -origin inclusion and k -origin equivalence are decidable for the class TDTT.*

Proof. Let $\mathcal{T}_1, \mathcal{T}_2$ be TDTTs over the same input and output alphabet. If $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$, then $(t, s, o) \in R_o(\mathcal{T}_1)$ implies that $(t, s, o) \in_k R_o(\mathcal{T}_1)$ for all $(t, s, o) \in R_o(\mathcal{T}_1)$. Lemma 13 yields that there are PTAs $\mathcal{A}_1, \mathcal{A}_2$ that recognize $\{H_{\mathcal{T}_1,0} \hat{\tau} \hat{s} \mid (t, s, o) \in R_o(\mathcal{T}_1)\}$ and $\{H_{\mathcal{T}_2,k} \hat{\tau} \hat{s} \mid (t, s, o) \in_k R_o(\mathcal{T}_2)\}$, respectively. Basically, we want to check that $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. However, we have to overcome a slight technical difficulty. If there are trees $H_{\mathcal{T}_1,0} \hat{\tau}_1 \hat{s}_1 \in L(\mathcal{A}_1)$ and $H_{\mathcal{T}_2,k} \hat{\tau}_2 \hat{s}_2 \in L(\mathcal{A}_2)$ such that for their encoded triples (t_1, s_1, o_1) and (t_2, s_2, o_2) holds that $t_1 = t_2$, $s_1 = s_2$ and o_1 and o_2 have an origin difference of at most k , i.e., $(t_1, s_1, o_1) \in_k \mathcal{R}_0(\mathcal{T}_2)$, it not necessarily holds that $H_{\mathcal{T}_1,0} \hat{\tau}_1 \hat{s}_1 \in L(\mathcal{A}_2)$. This is due to the fact that the base trees $H_{\mathcal{T}_1,0}$ and $H_{\mathcal{T}_2,k}$ look different in general because choices for Out in the first tree are based on the rules of \mathcal{T}_1 and without origin distance and in the latter tree based on the rules of \mathcal{T}_2 with k -origin distance. We only care whether the paths reachable by following the annotations τ_1 and s_1 through $H_{\mathcal{T}_1,0}$ and the paths reachability by following the annotations τ_2 and s_2 through $H_{\mathcal{T}_2,k}$ are the same. Thus, we introduce the operation *purge* which applied to a tree annotated with strategies of In and Out removes all non-strategy paths. It is not difficulty to see that the sets $L_1 := \{\text{purge}(H_{\mathcal{T}_1,0} \hat{\tau} \hat{s}) \mid (t, s, o) \in R_o(\mathcal{T}_1)\}$ and $L_2 := \{\text{purge}(H_{\mathcal{T}_2,k} \hat{\tau} \hat{s}) \mid (t, s, o) \in_k R_o(\mathcal{T}_2)\}$ are also PTA-recognizable. Hence, in order to check whether $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$, we have to check whether $L_1 \subseteq L_2$, which is decidable. We have shown that k -origin inclusion for TDTTs is decidable, consequently, k -origin equivalence for TDTTs is decidable for all $k \geq 0$. ◀

We show that checking whether a TDTT is k -origin DTDTT-uniformizable reduces to deciding emptiness of PTAs.

► **Proposition 15.** *Given $k \geq 0$, k -origin DTDTT-uniformization is decidable for the class TDTT.*

Proof. Given a TDTT \mathcal{T} , by Lemma 13, there is a PTA that recognizes

$$\{H_{\mathcal{T},k} \hat{\tau} \hat{s} \mid (t, s, o) \in_k R_o(\mathcal{T})\}.$$

By closure under complementation and intersection, there is a PTA that recognizes

$$\{H_{\mathcal{T},k} \hat{\tau} \hat{s} \mid (t, s, o) \notin_k R_o(\mathcal{T})\}.$$

By closure under projection, there is a PTA that recognizes

$$\{\mathsf{H}_{\mathcal{T},k} \hat{\ } \mathsf{s} \mid \exists \mathbf{t} : (t, s, o) \notin_k R_o(\mathcal{T})\}.$$

By closure under complementation and intersection, there is a PTA that recognizes

$$\{\mathsf{H}_{\mathcal{T},k} \hat{\ } \mathsf{s} \mid \forall \mathbf{t} : (t, s, o) \in_k R_o(\mathcal{T})\}.$$

By closure under projection, there is a PTA that recognizes

$$\{\mathsf{H}_{\mathcal{T},k} \mid \exists \mathbf{s} : \forall \mathbf{t} : (t, s, o) \in_k R_o(\mathcal{T})\}.$$

Let \mathcal{A} denote the PTA obtained in the last construction step. We show that \mathcal{T} is k -origin DTDTT-uniformizable iff $L(\mathcal{A}) \neq \emptyset$. We have that $L(\mathcal{A}) = \{\mathsf{H}_{\mathcal{T},k} \mid \exists \mathbf{s} : \forall \mathbf{t} : (t, s, o) \in_k R_o(\mathcal{T})\}$. Colloquially, this means that we can fix output choices that only depend on the previously seen input choices, which exactly describes DTDTT-uniformizability.

Assume \mathcal{T} is k -origin DTDTT-uniformizable, say by a DTDTT \mathcal{T}' . There exists a strategy of Out in $\mathsf{H}_{\mathcal{T},k}$ that copies the computations of \mathcal{T}' . Clearly, since \mathcal{T}' is deterministic, we obtain that $\exists \mathbf{s} : \forall \mathbf{t} : (t, s, o) \in_k R_o(\mathcal{T})$, \mathbf{s} can be chosen to be the strategy that copies \mathcal{T}' . Thus, $L(\mathcal{A}) \neq \emptyset$.

For the other direction, assume that $L(\mathcal{A}) \neq \emptyset$. This implies that also the set $\{\mathsf{H}_{\mathcal{T},k} \hat{\ } \mathsf{s} \mid \forall \mathbf{t} : (t, s, o) \in_k R_o(\mathcal{T})\}$ is non-empty and PTA recognizable. Since the set is PTA recognizable, it contains a regular infinite tree (meaning the tree has a finite representation). This tree implicitly contains a finite representation of some strategy \mathbf{s} such that $\forall \mathbf{t} : (t, s, o) \in_k R_o(\mathcal{T})$. Hence, the strategy \mathbf{s} can be translated into a finite-state DTDTT that k -origin uniformizes \mathcal{T} . \blacktriangleleft

Finally, combining Propositions 14 and 15, we obtain our main result.

► **Theorem 16.** *Given $k \geq 0$, k -origin inclusion, k -origin equivalence, and k -origin DTDTT-uniformization are decidable for the class TDDT.*

5 Conclusion

We introduced two similarity measures for TDDTs based on their behavior and studied decision problems for similar TDDTs. For TDDTs with bounded delay, the decision problems remain undecidable. For origin-close TDDTs they become decidable. For future work, we plan to consider other tree transducer models. In [16], it was shown that origin inclusion and origin equivalence are decidable for MSO tree transducers and macro tree transducers.

References

- 1 Mikolaj Bojanczyk. Transducers with origin information. In *ICALP (2)*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014.
- 2 Mikolaj Bojanczyk, Laure Daviaud, Bruno Guillon, and Vincent Penelle. Which classes of origin graphs are generated by transducers. In *ICALP*, volume 80 of *LIPICs*, pages 114:1–114:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 3 Sougata Bose, Shankara Narayanan Krishna, Anca Muscholl, Vincent Penelle, and Gabriele Puppis. On synthesis of resynchronizers for transducers. In *MFCS*, volume 138 of *LIPICs*, pages 69:1–69:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- 4 Sougata Bose, Anca Muscholl, Vincent Penelle, and Gabriele Puppis. Origin-equivalence of two-way word transducers is in PSPACE. In *FSTTCS*, volume 122 of *LIPICs*, pages 22:1–22:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
- 5 Fabienne Braune, Nina Seemann, Daniel Quernheim, and Andreas Maletti. Shallow local multi-bottom-up tree transducers in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 811–821, 2013.

- 6 J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. doi:10.1090/S0002-9947-1969-0280205-0.
- 7 Arnaud Carayol and Christof Löding. Uniformization in Automata Theory. In *Proceedings of the 14th Congress of Logic, Methodology and Philosophy of Science Nancy, July 19-26, 2011*, pages 153–178. London: College Publications, 2014.
- 8 Alonzo Church. Logic, arithmetic and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962.
- 9 María Emilia Descotte, Diego Figueira, and Santiago Figueira. Closure properties of synchronized relations. In *STACS*, volume 126 of *LIPICs*, pages 22:1–22:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- 10 Joost Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Syst. Theory*, 10:289–303, 1977.
- 11 Joost Engelfriet and Sebastian Maneth. Macro tree translations of linear size increase are mso definable. *SIAM Journal on Computing*, 32(4):950–1006, 2003.
- 12 Joost Engelfriet, Sebastian Maneth, and Helmut Seidl. Look-ahead removal for total deterministic top-down tree transducers. *Theor. Comput. Sci.*, 616:18–58, 2016.
- 13 Zoltán Ésik. Decidability results concerning tree transducers i. *Acta Cybernetica*, 5(1):1–20, 1980.
- 14 Diego Figueira and Leonid Libkin. Synchronizing relations on words. *Theory Comput. Syst.*, 57(2):287–318, 2015.
- 15 Emmanuel Filiot, Ismaël Jecker, Christof Löding, and Sarah Winter. On equivalence and uniformisation problems for finite transducers. In *ICALP*, volume 55 of *LIPICs*, pages 125:1–125:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.
- 16 Emmanuel Filiot, Sebastian Maneth, Pierre-Alain Reynier, and Jean-Marc Talbot. Decision problems of tree transducers with origin. *Inf. Comput.*, 261:311–335, 2018.
- 17 Patrick C. Fischer and Arnold L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2(1):88–101, 1968. doi:10.1016/S0022-0000(68)80006-6.
- 18 Zoltán Fülöp and Heiko Vogler. *Syntax-directed semantics: Formal models based on tree transducers*. Springer Science & Business Media, 2012.
- 19 Timothy V. Griffiths. The unsolvability of the equivalence problem for lambda-free nondeterministic generalized machines. *Journal of the ACM*, 15(3):409–413, 1968.
- 20 Shizuya Hakuta, Sebastian Maneth, Keisuke Nakano, and Hideya Iwasaki. Xquery streaming by forest transducers. In *2014 IEEE 30th International Conference on Data Engineering*, pages 952–963. IEEE, 2014.
- 21 Michael Holtmann, Łukasz Kaiser, and Wolfgang Thomas. Degrees of lookahead in regular infinite games. In *Foundations of Software Science and Computational Structures*, volume 6014 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2010. doi:10.1007/978-3-642-12032-9_18.
- 22 Frederick A. Hosch and Lawrence H. Landweber. Finite delay solutions for sequential conditions. In *ICALP*, pages 45–60, 1972.
- 23 Ralf Küsters and Thomas Wilke. Transducer-based analysis of cryptographic protocols. *Information and Computation*, 205(12):1741–1776, 2007.
- 24 Christof Löding and Sarah Winter. Uniformization problems for tree-automatic relations and top-down tree transducers. In *MFCS*, volume 58 of *LIPICs*, pages 65:1–65:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.
- 25 Christof Löding and Sarah Winter. Synthesis of deterministic top-down tree transducers from automatic tree relations. *Inf. Comput.*, 253:336–354, 2017.
- 26 Andreas Maletti. Tree transformations and dependencies. In *Conference on Mathematics of Language*, pages 1–20. Springer, 2011.
- 27 Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. The power of extended top-down tree transducers. *SIAM Journal on Computing*, 39(2):410–430, 2009.

- 28 Kazutaka Matsuda, Kazuhiro Inaba, and Keisuke Nakano. Polynomial-time inverse computation for accumulative functions with multiple data traversals. *Higher-Order and Symbolic Computation*, 25(1):3–38, 2012.
- 29 Tova Milo, Dan Suci, and Victor Vianu. Typechecking for xml transformers. *J. Comput. Syst. Sci.*, 66(1):66–97, 2003. doi:10.1016/S0022-0000(02)00030-2.
- 30 William C Rounds. Mappings and grammars on trees. *Mathematical systems theory*, 4(3):257–287, 1970.
- 31 James W. Thatcher. Generalized² sequential machine maps. *Journal of Computer and System Sciences*, 4(4):339–367, 1970.
- 32 Arie Van Deursen, Paul Klint, and Frank Tip. Origin tracking. *Journal of Symbolic Computation*, 15(5-6):523–545, 1993.
- 33 Janis Voigtländer and Armin Kühnemann. Composition of functions with accumulating parameters. *Journal of functional programming*, 14(3):317, 2004.
- 34 Sarah Winter. Uniformization problems for synchronizations of automatic relations on words. In *ICALP*, volume 107 of *LIPICs*, pages 142:1–142:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

A

 Missing example

► **Example 17.** To begin with, consider the FSTs $\mathcal{T}_1, \mathcal{T}_2$ depicted in Figure 4a, we already explained in the proof of Proposition 7 that $\mathcal{T}_1 = \mathcal{T}_2$, but $\mathcal{T}_1 \neq_k \mathcal{T}_2$ for all $k \geq 0$, i.e., their origin distance is unbounded. However, their delay is bounded by 1. It is easy to see that $\mathcal{T}_1 =_{\mathbb{D}_1} \mathcal{T}_2$, because their difference in the length of their outputs for the same input is at most one letter. Now, consider the FSTs $\mathcal{T}_3, \mathcal{T}_4$ depicted in Figure 4b. Both recognize the relation $\{(ab, c^n) \mid n \in \mathbb{N}\}$, hence, $\mathcal{T}_3 = \mathcal{T}_4$. Clearly, their origin distance is bounded by 1. The whole output either has the first or the second letter as origin. However, $\mathcal{T}_3 \neq_{\mathbb{D}_k} \mathcal{T}_4$ for all $k \geq 0$, i.e., their delay is unbounded. For any k , take the consider the unique runs that admit output c^{k+1} in \mathcal{T}_3 and \mathcal{T}_4 , respectively. We compare these runs for the input prefix a , \mathcal{T}_3 already has produced c^{k+1} , and \mathcal{T}_4 no output so far. Their delay is $k + 1$.

B

 Missing proofs of Propositions 7 and 8

► **Proposition 7.**

1. There exist FSTs $\mathcal{T}_1, \mathcal{T}_2$ such that $\mathcal{T}_1 \subseteq \mathcal{T}_2$, but $\mathcal{T}_1 \not\subseteq_k \mathcal{T}_2$ for all $k \geq 0$.
2. There exist FSTs $\mathcal{T}_1, \mathcal{T}_2$ such that $\mathcal{T}_1 = \mathcal{T}_2$, but $\mathcal{T}_1 \neq_k \mathcal{T}_2$ for all $k \geq 0$.
3. There exists an FST \mathcal{T} such that \mathcal{T} is sequentially uniformizable, but \mathcal{T} is not k -origin sequentially uniformizable for all $k \geq 0$.

Proof. Secondly, consider the FSTs $\mathcal{T}, \mathcal{T}'$ depicted in Figure 4a. The recognized relation $\mathcal{R}(\mathcal{T}) \subseteq \{a, b, A, B\}^* \times \{a, b\}^*$ consists of $\{(ab^n A, ab^n) \mid n \in \mathbb{N}\}$ and $\{(ab^n B, ab^m) \mid 0 \leq m \leq 2n - 1, n \in \mathbb{N}\}$. The sequential transducer \mathcal{T}' recognizes the function $f: \{a, b, A, B\}^* \rightarrow \{a, b\}^*$ defined by $f(ab^n X) = ab^n$ for $X \in \{A, B\}$ and all $n \in \mathbb{N}$. Clearly, \mathcal{T}' is a sequential uniformization of \mathcal{T} . However, no sequential uniformization with bounded origin distance exists. Towards a contradiction, assume there is sequential transducer \mathcal{T}'' that uniformizes \mathcal{T} such that $\mathcal{T}'' \subseteq_k \mathcal{T}$ for some $k \geq 0$. Consider the input word $ab^{2k} A$, in \mathcal{T} there is only one run with the input which yields the output ab^{2k} and the origin of the i th output letter is the i th input letter for all i . Since $\mathcal{T}'' \subseteq_k \mathcal{T}$ there exists a run of \mathcal{T}'' on $ab^{2k} A$ that yields ab^{2k} and the origin of the first b in the output is at latest the k th b in the input. Now, consider the input $ab^{6k} B$, the output of \mathcal{T}'' on $ab^{6k} B$ is ab^{6k} . Since \mathcal{T}'' is sequential, the the runs of \mathcal{T}'' on $ab^{2k} A$ and $ab^{6k} B$ are the same up to the input ab^{2k} , thus, also for the output ab^{6k}

the origin first output b is at latest the k th b in the input. Now we compare this with all possible runs in \mathcal{T} on $ab^{6k}B$ that also yield ab^{6k} . Note that \mathcal{T} (after producing the first b) must always produce two b at once, thus in order to produce ab^{6k} for the input $ab^{6k}B$, the production of b can only start after while reading the second half of the input. This implies that the first output has an origin in the second half of input which has a distance of more than k (at least $2k$) to the k th b in the input. ◀

► **Proposition 8.** *Let $\mathcal{T}_1, \mathcal{T}_2$ be real-time FSTs, if $\mathcal{T}_1 \subseteq_i \mathcal{T}_2$ for some $i \geq 0$, then $\mathcal{T}_1 \subseteq_{\mathbb{D}_j} \mathcal{T}_2$ for some $j \geq 0$.*

Proof. Let $\mathcal{T}_1, \mathcal{T}_2$ be real-time FSTs such that $\mathcal{T}_1 \subseteq_i \mathcal{T}_2$ for some $i \geq 0$. Let ℓ be the maximum number of output letters that \mathcal{T}_1 produces in a computation step. Consider any $(u, v, o_1) \in R_o(\mathcal{T}_1)$, since $\mathcal{T}_1 \subseteq_i \mathcal{T}_2$, there is $(u, v, o_2) \in R_o(\mathcal{T}_2)$ such that $dist(o_1(d), o_2(d)) \leq i$ for all $d \in dom_v$. Let ρ_1 and ρ_2 be the corresponding runs of \mathcal{T}_1 and \mathcal{T}_2 , respectively. We show that $delay(\rho_1, \rho_2) \leq \ell \cdot i$ which implies that $\mathcal{T}_1 \subseteq_{\mathbb{D}_{\ell \cdot i}} \mathcal{T}_2$. Let $u = a_1 \cdots a_n$ and $v = b_1 \cdots b_m$. Pick any prefix of u , say $a_1 \cdots a_k$, and consider the prefixes of the runs ρ_1 and ρ_2 such that the input $a_1 \cdots a_k$ has been processed. Let $b_1 \cdots b_{k_1}$ and $b_1 \cdots b_{k_2}$ be the respective produced outputs. Wlog., let $k_1 \leq k_2$. If $k_1 = k_2$, then the output delay for the prefix $a_1 \cdots a_k$ is zero. So assume $k_1 < k_2$. We have to show that $|b_{k_1+1} \cdots b_{k_2}|$ is less than $\ell \cdot i$. Since the origin mappings of ρ_1 and ρ_2 , that is, o_1 and o_2 , have a distance of at most i , we know that the origin of $b_{k_1+1} \cdots b_{k_2}$ in ρ_1 is no later than at the letter a_{k+i} . On $a_{k+1} \cdots a_{k+i}$, \mathcal{T}_1 can produce at most $\ell \cdot i$ output letters. Consequently, $|b_{k_1+1} \cdots b_{k_2}| \leq \ell \cdot i$. ◀